明　細　書

キャッシュメモリおよびその制御方法

技術分野

本発明は、プロセッサのメモリアクセスを高速化するためのキャッシュメモリおよびその制御方法に関する。

背景技術

キャッシュメモリは、主記憶装置のアクセス時間を短縮しプロセッサの処理能力の向上を図るために、従来から広く用いられている。

例えば特開平６－２６６６２０号公報等に開示されたキャッシュメモリは、メインメモリのブロック単位のデータを各エントリに格納し、このエントリを介して、マルチタスク処理を行なう処理ユニットからのアクセスに対応するデータの転送制御と排他制御を行なう。このキャッシュメモリは、エントリに格納されているブロックを排他制御の対象として設定した処理ユニットの各タスクの識別情報を登録するタスク識別情報登録部を、エントリごとに設ける構成とし、タスク単位で、エントリに格納されているブロックの排他制御、および、この排他制御の設定と解除を行なっている。

このキャッシュメモリによれば、マルチタスク処理における排他制御を効率良く行ない、タスク間で共通に使用するデータの矛盾を解消することを図っている。

しかしながら、上記従来技術におけるキャッシュメモリによれば、プロセッサのタスク切り替えに伴ってキャッシュメモリのヒット率が実行中でない他のタスクによる影響を受けるという問

題がある。

例えば、タスクＡの命令列（又はデータ）がキャッシュメモリに格納されている状態でタスクＡの実行からタスクＢの実行に切り換えられた場合、タスクＢの実行によりキャッシュメモリ中のタスクＡの命令列（又はデータ）が追い出されてしまう。タスクＡの命令列（又はデータ）がキャッシュメモリから追い出されていれば、再度タスクＡが実行されたときに、キャッシュミスが発生するという問題がある。特に、圧縮音声データや圧縮映像データのデコード／エンコード処理などのリアルタイム性を必要とする処理では、上記タスク切り替えに伴う他のタスクの影響によって、タスク切り替え後のキャッシュのリプレース処理によってタスクの割当時間を侵食され、必要な処理時間を確保できず、リアルタイム性が損なわれるあるいは処理時間を確定できないという問題がある。

発明の開示

本発明は、タスク切り替え等によるキャッシュメモリの他のタスクの影響を防止し、タスクの実質的な処理時間を容易に確保するキャッシュメモリを提供することを目的とする。

上記目的を達成するため本発明のキャッシュメモリは、Ｎ－ウェイ・セット・アソシエイティブ方式のキャッシュメモリであって、Ｎ個のウェイのうち１つ以上のウェイを示す制御レジスタと、制御レジスタに示されるウェイをアクティブにする制御手段と、制御レジスタの内容を更新する更新手段とを備える。

この構成によれば、キャッシュメモリを構成するＮ－ウェイのうち、制御レジスタに示されたウェイのみをアクティブにし、し

かも制御レジスタの内容は更新可能なので、プロセッサが実行する処理に応じてアクティブなウェイを動的に設定することができる。タスクとウェイとを対応付ければ、タスク切り替え後に他のタスクによりキャッシュメモリから必要なデータが追い出されることが解消され、タスク切り替えに伴うヒット率の他のタスクからの影響を防止することができる。その結果、タスクに必要とされる実質的な処理時間を容易に確保することができる。

ここで、前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイ（インアクティブなウェイと呼ぶ。）に対して少なくともリプレースを制限する構成としてもよい。

この構成によれば、インアクティブなウェイについては、少なくともリプレースが制限される。つまり、インアクティブなウェイについて完全にディスエーブルにしても、リプレースだけをディスエーブルにしてもよい。後者の場合、キャッシュメモリに対するリード／ライトまでは制限されないので、ヒット率の低化を防止し、かつインアクティブなウェイを有効に活用することができる。

ここで、前記キャッシュメモリは、さらに、ウェイ毎に設けられ、キャッシュデータのアドレスをタグとして保持するタグ保持手段と、プロセッサから出力されるメモリアクセスアドレスの上位部分であるタグアドレスと、タグ保持手段から出力されるＮ個のタグとを比較することによりヒットかミスヒットかを判定するＮ個の比較手段を有し、前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイに対応する比較手段をディスエーブルにする構成としてもよい。

この構成によれば、インアクティブなウェイに対応する比較手

段をディスエーブルにするので、比較手段における消費電力を低減することができる。

ここで、前記制御手段は、さらに、制御レジスタに示されたアクティブなウェイ以外のウェイに対応するタグ保持手段に対して、比較手段へのタグ出力をディスエーブルにする構成としてもよい。

この構成によれば、インアクティブなウェイに対応するタグ出力と比較手段とがディスエーブルされるので、タグ保持手段の消費電力を低減することができる。

ここで、前記制御手段は、プロセッサからメモリアクセスアドレスが出力されたとき、当該アクセスアドレスについて、比較手段に最大２回のタグ比較を行わせるよう制御し、１回目のタグ比較では、制御レジスタに示されたアクティブなウェイ以外のウェイに対応する比較手段をディスエーブルし、１回目のタグ比較においてミスヒットと判定された場合に、アクティブなウェイ以外のウェイに対応する比較手段をディスエーブルしないで２回目のタグ比較を行わせる構成としてもよい。

この構成によれば、１回目のタグ比較におけるヒット率が高いほど、比較手段における消費電力を低減することができ、しかも、１回目のタグ比較においてミスヒットした場合に２回目のタグ比較を行うので、インアクティブなウェイのキャッシュデータも有効に活用することができる。

ここで、前記制御手段は、前記２回目のタグ比較においてアクティブなウェイに対応する比較手段をディスエーブルする構成としてもよい。

この構成によれば、２回目のタグ比較ではインアクティブなウ

ェイに対応する比較手段のみがタグ比較を行うので、さらに消費電力を低減することができる。

ここで、前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイに対して、その状態の更新を禁止する構成としてもよい。

この構成によれば、例えばインアクティブなウェイの状態を示すフラグ類の更新を禁止することにより、インアクティブなウェイに対するタスク切り替えによる影響を防止することができる。

ここで、前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイについて、そのアクセス順序を示す情報の更新を禁止する構成としてもよい。

この構成によれば、アクセス順序を示す情報の更新を禁止するので、更新手段による更新によってインアクティブなウェイがアクティブなウェイに割り当てられたタスクのキャッシュミス時のリプレース順が変化することがない。

ここで、前記キャッシュメモリは、さらに、前記更新手段によって制御レジスタの内容が更新されたとき、ウェイに対するアクセス順序を示す情報をリセットするリセット手段を有する構成としてもよい。また、前記アクセス順序を示す情報は、キャッシュエントリー毎の１ビットデータであり、前記キャッシュメモリは、さらに、リプレース可能な複数ウェイから１つのウェイをラウンドロビン方式で選択するためラウンド位置を示すデータを保持するレジスタを有し、前記リセット手段は、前記更新手段によって制御レジスタの内容が更新されたとき、前記レジスタをリセットするように構成してもよい。

この構成によれば、インアクティブなウェイを割り当てたタス

クのキャッシュミス時のリプレース順への影響をなくすことができる。

ここで、前記更新手段は、アクティブにすべきウェイを指定するウェイデータであって、タスク毎のウェイデータを保持する保持手段と、実行中のタスクに対応するウェイデータを保持するよう前記制御レジスタを書き換える書き換え手段とを有する構成としてもよい。

この構成によれば、タスクが切り替わる毎に動的に、制御レジスタを書き換えるので、他タスク毎にアクティブなウェイを対応付けることができる。

ここで、前記保持手段は、メモリ中に記憶されたタスク毎のコンテキストデータの一部として前記ウェイデータを保持し、前記書き換え手段は、タスク切り替えに際して、制御レジスタ中の現タスクのウェイデータをメモリに退避し、次タスクのウェイデータをメモリから前記制御レジスタに復帰する構成としてもよい。

この構成によれば、制御レジスタの更新は、ＯＳ（Operating System）によるタスク切り替えにより、キャッシュメモリのハードウェアを大幅に追加することなく簡単に実現することができる。

ここで、前記保持手段は、タスク毎の前記ウェイデータを保持し、前記書き換え手段は、メモリに記憶された各タスクのアドレス範囲を記憶するアドレス記憶手段と、アドレス記憶手段に記憶されたアドレス範囲と、プロセッサから出力される命令フェッチアドレスとに基づいて、実行中のタスクを判別する判別手段と、判別された実行中のタスクに対応するウェイデータを前記保持手段から選択する選択手段と、選択されたウェイデータを前記制

御レジスタに書き込む書き込み手段とを備える構成としてもよい。

この構成によれば、制御レジスタの更新は、キャッシュメモリ自身が主体的に判断することにより行われるので、どのようなプロセッサに対しても、タスク毎に対応するウェイをアクティブにすることができる。

ここで、前記保持手段は、タスク毎の前記ウェイデータを保持し、前記書き換え手段は、プロセッサから出力されるタスク番号に従って、実行中のタスクに対応するウェイデータを前記保持手段から選択する選択手段と、選択されたウェイデータを前記制御レジスタに書き込む書き込み手段とを備える構成としてもよい。

この構成によれば、プロセッサから出力されるタスク番号を利用するのでハードウェアを大幅に追加することなく、制御レジスタを簡単に更新し、タスク毎に対応するウェイをアクティブにすることができる。

ここで、前記保持手段に保持されるウェイデータは、ＯＳによってタスクに割り当てられるように構成してもよい。

この構成によれば、タスクへのウェイの割り当てをＯＳが行うことによって、各タスクへのウェイの割り当てを最適化することが容易になる。

ここで、前記キャッシュメモリは、各ウェイにおけるリプレース単位をキャッシュエントリーのラインサイズと、ラインサイズの２のｎ乗分の１のサイズとに切り替え可能であり、前記制御レジスタは、さらに、ウェイ毎のリプレースサイズを示し、前記制御手段は、制御レジスタに示されたリプレースサイズを単位としてリプレース制御を行う構成としてもよい。

また、前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイに対して少なくともリプレースを制限し、制御レジスタに示されたアクティブなウェイに対して制御レジスタに示されたサイズを単位にリプレースを行う構成としてもよい。

ここで、前記更新手段は、アクティブにすべきウェイを指定するウェイデータであってタスク毎のウェイデータと、タスク毎のリプレースサイズとを保持する保持手段と、実行中のタスクに対応するウェイデータ及びリプレースサイズを保持するよう前記制御レジスタを書き換える書き換え手段とを有する構成としてもよい。

この構成によれば、タスク毎にアクティブなウェイを切り換えると同時に、リプレース単位をも切り換えることができるので、タスクの処理内容に応じてミスヒットを低減することができる。

また、本発明のキャッシュメモリの制御方法は、Ｎ－ウェイ・セット・アソシエイティブ方式のキャッシュメモリを制御する制御方法であって、Ｎ個のウェイのうち１つ以上のウェイを示すウェイデータを制御レジスタに設定するステップと、制御レジスタに示されるウェイをアクティブにする制御ステップとを有する。

ここで、前記制御ステップでは、制御レジスタに示されたアクティブなウェイ以外のウェイに対して少なくともリプレースを制限するようにしてもよい。

ここで、前記制御方法は、さらに、アクティブにすべきウェイを指定するウェイデータであってタスク毎のウェイデータを保持する保持部から、実行中のタスクに対応するウェイデータ読み出して、読み出したウェイデータを前記制御レジスタに書き込む

更新ステップを有する構成としてもよい。

以上説明したように、本発明のキャッシュメモリによれば、プロセッサが実行する処理毎にアクティブなウェイを動的に設定することができるので、タスクとウェイとを対応付ければ、タスク切り替え後に他のタスクによりキャッシュメモリから必要なデータが追い出ることが解消され、タスク切り替えに伴うヒット率の他のタスクからの影響を防止することができる。その結果、タスクに必要とされる実質的な処理時間を容易に確保することができる。

図面の簡単な説明

図１は、本発明の実施の形態１におけるプロセッサ、キャッシュメモリ、メモリを含むシステムの概略構成を示すブロック図である。

図２は、キャッシュメモリの構成例を示すブロック図である。

図３は、ウェイ・レジスタのビット構成を示す図である。

図４は、ウェイ・レジスタとウェイとの対応関係を示す説明図である。

図５は、制御部におけるリプレース処理を示すフローチャートである。

図６は、タスク切り替え処理を示すフローチャートである。

図７は、タスク領域のコンテキストに含まれるタスク毎のウェイデータを示す図である。

図８は、本発明の実施の形態２におけるキャッシュメモリの構成を示すブロック図である。

図９は、キャッシュエントリーのビット構成を示す図である。

図１０は、設定部の構成を示すブロック図である。

図１１は、フラグの更新例を示す説明図である。

図１２は、フラグ更新処理フローを示す図である。

図１３は、リプレース処理フローを示す図である。

図１４は、本発明の実施の形態３におけるキャッシュメモリの構成を示すブロック図である。

図１５は、キャッシュエントリーのビット構成を示す図である。

図１６は、リプレースサイズレジスタのビット構成を示す図である。

図１７は、１２ベース　フラグの更新例を示す説明図である。

図１８は、ベース　リプレース処理を示すフローチャートである。

図１９は、本発明の実施の形態４における比較制御部および各ウェイの要部の構成を示すブロック図である。

図２０は、イネーブル回路の制御論理を示す真理値表をである。

発明を実施するための最良の形態

（実施の形態１）

図１は、本発明の実施の形態１におけるプロセッサ１、キャッシュメモリ３、メモリ２を含むシステムの概略構成を示すブロック図である。同図のように、本発明のキャッシュメモリ３は、プロセッサ１およびメモリ２を有するシステムに備えられる。プロセッサ１は、マルチタスク制御を行うプロセッサであり、メモリ２中のタスク１～４等を切り替えて実行する。キャッシュメモリ３は、Ｎ－ウェイ・セット・アソシエイティブ方式のキャッシュメモリであって、タスク毎にＮ個のウェイのうち１つ以上のウェ

イを対応させて、当該タスク実行中に対応するウェイをアクティブにするよう構成されている。各タスクは、アクティブなウェイに対しては、キャッシュメモリとしての全機能を利用可能である。また、各タスクは、アクティブでないウェイ（インアクティブなウェイと呼ぶ）に対しては、キャッシュメモリとしての全機能のうち利用可能な機能が制限されている。本実施の形態では、インアクティブなウェイは、リプレースする機能が制限され、それ以外のリード、ライト等は制限されていないものとする。

以下では、キャッシュメモリ３の具体例として、８ウェイ・セット・アソシエイティブ方式のキャッシュメモリに本発明を適用した場合の構成について説明する。

図２は、キャッシュメモリ３の構成例を示すブロック図である。同図のように、キャッシュメモリ３は、アドレスレジスタ２０、デコーダ３０、８つのウェイ３１ａ～３１ｈ（以下ウェイ０～７と略す）、８つの比較器３２ａ～３２ｈ、８つのアンド回路３３ａ～３３ｈ、オア回路３４、セレクタ３５、セレクタ３６、デマルチプレクサ３７、制御部３８を備える。

アドレスレジスタ２０は、メモリ２へのアクセスアドレスを保持するレジスタである。このアクセスアドレスは３２ビットであるものとする。同図に示すように、アクセスアドレスは、最上位ビットから順に、２１ビットのタグアドレス、４ビットのセットインデックス（図中のＳＩ）、５ビットのワードインデックス（図中のＷＩ）を含む。

ここで、タグアドレスはウェイにマッピングされるメモリ中の領域（そのサイズはセット数×ブロックである。）を指す。この領域のサイズは、タグアドレスよりも下位のアドレスビット（Ａ

１０～Ａ０）で定まるサイズつまり２ｋバイトであり、１つのウェイのサイズでもある。セットインデックス（ＳＩ）はウェイ０～３に跨る複数セットの１つを指す。このセット数は、セットインデックスが４ビットなので１６セットである。タグアドレスおよびセットインデックスで特定されるブロックは、リプレース単位であり、キャッシュメモリに格納されている場合はラインデータ又はラインと呼ばれる。ラインデータのサイズは、セットインデックスよりも下位のアドレスビットで定まるサイズつまり１２８バイトである。１ワードを４バイトとすると、１ラインデータは３２ワードである。ワードインデックス（ＷＩ）は、ラインデータを構成する複数ワード中の１ワードを指す。アドレスレジスタ２０中の最下位２ビット（Ａ１、Ａ０）は、ワードアクセス時には無視される。

デコーダ３０は、セットインデックスの４ビットをデコードし、８つのウェイ０～７の同順に位置するキャッシュエントリーからなる１６個のセット中の１つのセットを選択する。

ウェイ０～７は、同じ構成を有数する８つのウェイであり、８×２ｋバイトの容量を有する。ウェイ０は、１６個のキャッシュエントリーを有する。１つのキャッシュエントリーは、バリッドフラグＶ、２１ビットのタグ、１２８バイトのラインデータを保持する。バリッドフラグＶは、そのキャッシュエントリーが有効か否かを示す。タグは２１ビットのタグアドレスのコピーである。ラインデータは、タグアドレスおよびセットインデックスにより特定されるブロック中の１２８バイトデータのコピーである。また、ウェイ１～７についても、ウェイ０と同様である。セットインデックスの４ビットによってデコーダ３０を介して選択され

る４ウェイに跨る４つのキャッシュエントリーは、セットと呼ばれる。また、同図では書き込みがあったことを示すダーティフラグは省略されている。

比較器３２ａは、アドレスレジスタ２０中のタグアドレスと、セットインデックスにより選択されたセットに含まれる４つのタグ中のウェイ０のタグとが一致するか否かを比較する。比較器３２ｂ～３２ｈについても、ウェイ３１ｂ～３１ｈに対応すること以外は同様である。

アンド回路３３ａは、バリッドフラグと比較器３２ａの比較結果とが一致するか否かを比較する。この比較結果をｈ０とする。比較結果ｈ０が１である場合は、アドレスレジスタ２０中のタグアドレスおよびセットインデックスに対応するラインデータが存在すること、つまりウェイ０においてヒットしたことを意味する。比較結果ｈ０が０である場合は、ミスヒットしたことを意味する。アンド回路３３ｂ～３３ｈについても、ウェイ３１ｂ～３１ｈに対応すること以外は同様である。その比較結果ｈ１～ｈ７は、ウェイ１～７でヒットしたかミスしたかを意味する。

オア回路３４は、比較結果ｈ０～ｈ３のオアをとる。このオアの結果をｈｉｔとする。ｈｉｔは、キャッシュメモリにヒットしたか否かを示す。

セレクタ３５は、選択されたセットにおけるウェイ０～７のラインデータのうち、ヒットしたウェイのラインデータを選択する。

セレクタ３６は、セレクタ３５により選択された３２ワードのラインデータにうち、ワードインデックスに示される１ワードを選択する。

デマルチプレクサ３７は、キャッシュエントリーにデータを書

き込む際に、ウェイ０～７の１つに書き込みデータを出力する。この書き込みデータはワード単位でよい。

制御部３８は、内部にウェイ・レジスタ３７１を有し、キャッシュメモリ３の全体の制御を行う。ウェイ・レジスタ３７１は、ウェイ０～７のうちアクティブなウェイを示すデータを保持するレジスタである。制御部３８は、ウェイ・レジスタ３７１によって示されるアクティブなウェイに対しては、キャッシュメモリとしての全機能を制限なく制御し、インアクティブなウェイに対しては、リプレースする機能を制限する。

図３は、ウェイ・レジスタ３７１のビット構成を示す図である。同図のように、ウェイ・レジスタ３７１は、３２ビットレジスタであり、下位８ビットにウェイ０～７に対応するＷ０フラグ～Ｗ７フラグを保持する。例えば、Ｗ０フラグが１のときウェイ０がアクティブなウェイであることを示し、０のときウェイ０がインアクティブなウェイであることを示す。Ｗ１フラグ～Ｗ７フラグについても同様である。以下、Ｗ０フラグ～Ｗ７フラグの集まりをアクティブウェイデータと呼ぶ。このウェイ・レジスタ３７１は、プロセッサ１から直接読み書き可能であり、各タスクのコンテキストの一部をなす。つまり、タスク毎にアクティブウェイデータを有し、タスク切り替えによって、ウェイ・レジスタ３７１の内容は実行中のタスクに対応するアクティブウェイデータに書き換えられる。

図４は、ウェイ・レジスタ３７１とウェイとの対応関係を示す説明図である。同図左側では、ウェイ・レジスタ３７１に保持されているアクティブウェイデータが”００１１１０００”であるので、ウェイ２、３、４がアクティブウェイとなり、ウェイ０、

１、５、６、７がインアクティブになる。タスク切り替えに際して、ウェイ・レジスタ３７１は、例えば同図右側のようなアクティブウェイデータに書き換えられる。同図右側では、ウェイ５～７がアクティブとなり、ウェイ０～４がインアクティブになる。

図５は、制御部３８におけるリプレース処理を示すフローチャートである。同図において、制御部３８は、ミスヒットが発生したか否かを判定し（Ｓ５１）、ミスヒットが発生したと判定された場合に、セットインデックスにより選択されたセットにおける、４つウェイのキャッシュエントリーの中からリプレース対象を１つ選択する（ステップＳ５２）。このリプレース対象の選択はＬＲＵ方式でよい。

さらに、制御部３８は、ウェイ・レジスタ３７１を参照して、選択されたウェイがアクティブであるか否かを判定し（Ｓ５３）、アクティブでなければステップＳ５２に戻り再度他のウェイのキャッシュエントリーを選択する。制御部３８は選択されたアクティブなウェイのキャシュエントリーをリプレースする（Ｓ５４）。

このように、制御部３８は、ウェイ・レジスタ３７１に示されるインアクティブなウェイに対しては、リプレースを制限し、制御部３８は、アクティブなウェイに対しては、リプレースを制限することなくキャッシュメモリとしても全機能を制御する。ここでは、リプレースの制限はリプレースの禁止としている。

図６は、プロセッサ１におけるタスク切り替え処理を示すフローチャートである。タスク切り替え処理は時間の経過やイベントの発生により起動される。同図においてプロセッサ１は、現在実行中のタスクのコンテキストをメモリ２中の例えばスタック領

域に退避し（ステップＳ６１）、次に実行すべきタスクのコンテキストをスタック領域から復帰させる（ステップＳ６２）。ここで、スタック領域は、図７に示すように、メモリ２に確保され、各タスクのコンテキストを記憶するための領域である。各タスクのコンテキストは、プロセッサの汎用レジスタのデータや、種々の制御レジスタのデータを含み、加えて、本実施の形態ではウェイ・レジスタに格納されるアクティブウェイデータを含む。

このようにして、ウェイ・レジスタ３７１は、タスク切り替えに際して書き換えられるので、常に実行中のタスクに対応するアクティブウェイデータを保持することになる。

以上説明してきたように、本実施の形態におけるキャッシュメモリによれば、プロセッサ１に実行されるタスクから見れば、キャッシュメモリへのリードおよびライトについてはアクティブなウェイに対してもインアクティブなウェイに対しても可能であるが、ミスヒットした場合にリプレース対象となるウェイについてはアクティブなウェイに制限されることになる。

例えば、図４において同図左側をタスク１実行時、右側をタスク２実行時のアクティブウェイとする。タスクの実行が経過するにつれて、タスク１のキャッシュデータは次第にウェイ２～３に格納されていき、タスク２のキャッシュデータは次第にウェイ４～７に格納されていくことになる。言い換えれば、ウェイ２～３に格納されたタスク１のキャッシュデータはタスク２の実行によって追い出されない（リプレースされない）。また、ウェイ４～７に格納されたタスク２のキャッシュデータはタスク１の実行によって追い出されない（リプレースされない）。その結果、タスク切り替えに伴ってタスク１では必要なキャッシュデータ

が、他のタスクによってリプレースされ、再度タスク１実行時に追い出されたデータをキャッシュにリプレースすることも解消できる。その結果、タスク切り替えに伴うリプレースの発生を低減させることができ、他のタスクからの影響を抑えることができる。

＜変形例＞

なお、本発明のキャッシュメモリは、上記の実施形態の構成に限るものではなく、種々の変形が可能である。以下、変形例のいくつかについて説明する。

（１）複数のタスクと複数のウェイの対応関係については、１つのタスクに独占されるウェイと、複数のタスクに共用されるウェイとを混在させることができる。例えば、図４において、ウェイ２～４はタスク１が独占し、ウェイ５～７はタスク２が独占し、ウェイ０、１は他のタスクが共用するものとする。この場合、タスク１および２は、ウェイを独占するので、タスク切り替えによるキャッシュミスを低減し、リアルタイム性を要する処理に適している。タスク１および２以外のタスクは、リアルタイム性を要しない処理等に適している。

（２）上記実施の形態では、制御部３８は、アクティブなウェイについてはキャッシュメモリの全機能を制御し、インアクティブなウェイについてはリプレースを禁止しているが、これに限らない。

例えば、制御部３８は、ウェイ・レジスタ３７１に示されたインアクティブなウェイ対して、その状態の更新を禁止する構成としてもよい。例えば、ウェイの状態を表すフラグ類の更新を禁止することにより、インアクティブなウェイに対するタスク切り替

えによる影響を防止することができる。

また、制御部３８は、ウェイ・レジスタ３７１に示されたインアクティブなウェイについて、そのアクセス順序を示す情報の更新を禁止する構成としてもよい。これによれば、アクセス順序を示す情報の更新を禁止するので、他タスクの実行によりリプレース順が影響を受けることがなくなる。

あるいは、制御部３８は、インアクティブなウェイについて、全機能を禁止するようにしてもよい。この場合、インアクティブなウェイのタグ出力を禁止するよう出力イネーブル信号をディスエーブルにすればよい。こうすれば、インアクティブなウェイの消費電力を低減することができる。また、全機能を禁止する場合には、各タスクがウェイを共用することなく独占するように、タスクとウェイとを対応付けることが望ましい。こうすれば、メモリとキャッシュメモリとの間でデータに矛盾が生じることを防止することができる。

また、制御部３８は、リプレースの禁止に加えて、アクセスの順番を示すＬＲＵ用の順序データを更新しないように構成してもよい。

（３）ウェイ・レジスタ３７１の内容が更新されたとき、ＬＲＵ方式で用いられるアクセス順序情報をリセットする構成としてもよい。

（４）また、制御部３８は、リプレース禁止の代わりにリプレース回数を制限する構成としてもよいし、ウェイ中の特定のキャッシュエントリーに対するリプレースを禁止し、その他のキャッシュエントリーに対してはリプレースを行う構成としてもよい。

（５）上記実施の形態では、８ウェイ・セット・アソシエイティ

ブのキャッシュメモリを例に説明したが、ウェイ数は、４ウェイでも１６ウェイでもいくつでもよい。また、上記実施の形態では、セット数が１６である例を説明したが、セット数はいくつでもよい。

（６）上記実施の形態では、セット・アソシエイティブのキャッシュメモリを例に説明したが、フル・アソシエイティブ方式のキャッシュメモリであってもよい。フル・アソシエイティブ方式の場合、セットが１つのケースと考えることができる。

（７）図４に示したアクティブウェイデータは、ＯＳによってタスク毎に割り当てられるようにしてもよい。すなわち、ＯＳは実行対象のタスクを生成するときに、そのタスクに対してアクティブにすべきウェイを割り当て、割り当てた結果からそのタスクのアクティブウェイデータを生成する。さらに、ＯＳは、生成されたアクティブウェイデータを、図７に示すように当該タスクのコンテキストデータの一部に設定すればよい。

（８）複数のタスクがメモリデータを共有する場合、それらのタスクには全く同じウェイを共有すべきである。例えば、タスクＡとタスクＢがメモリデータを共有する場合、タスクＡにウェイ５、６、７が割り当てられていれば、ＯＳは、タスクＢにもウェイ５、６、７を割り当てればよい。

（実施の形態２）

実施の形態１では、ウェイ・レジスタ３７１をタスク切り替えによって書き換える構成を説明したが、本実施の形態では、キャッシュメモリにおいてタスクを判別して判別結果に応じてウェイ・レジスタ３７１を書き換える構成について説明する。加えて、実施の形態１ではリプレースアルゴリズムが周知のＬＲＵ方式

としたが、本実施の形態ではアクセス順序を示すデータの代わりに１ビットのフラグを用いる擬似的なＬＲＵ方式を行う構成について説明する。

図８は、本発明の実施の形態２におけるキャッシュメモリの構成を示すブロック図である。同図のキャッシュメメリは、図２の構成と比較して、ウェイ３１ａ～３１ｄの代わりにウェイ１３１ａ～１３１ｄを備える点と、制御部３８の代わりに制御部１３８を備える点とが異なっている。以下、同じ点は説明を省略して、異なる点を中心に説明する。

ウェイ１３１ａは、ウェイ３１ａと比べて、各キャッシュエントリー中に、使用フラグとニューフラグとが追加されている点が異なる。図９に、キャッシュエントリーのビット構成を示す。１つのキャッシュエントリーは、バリッドフラグＶ、２１ビットのタグ、１２８バイトのラインデータ、使用フラグＵ、ニューフラグＮおよびダーティフラグＤを保持する。このうち、使用フラグＵは、そのキャッシュエントリーにアクセスがあったか否かを示し、ミスヒットによるリプレースに際してセット内の８つのキャッシュエントリーにおけるアクセス順序の代わりに用いられる。より正確には、使用フラグＵの１は、アクセスがあったことを、０はないことを意味する。セット内の８つの使用フラグは、全て１になれば、０にリセットされるので、セット内の８つのキャッシュエントリーにおける使用の有無を示す相対的な値である。別言すれば、使用フラグＵは、アクセスされた時期が古いか新しいか２つの相対的な状態を示す。つまり、使用フラグＵが１のキャッシュエントリーは、使用フラグが０のキャッシュエントリーよりも新しくアクセスされたことを意味する。また、ニューフラグ

Ｎは、リプレース直後（又はフィル直後）に初期値として１が設定され、当該キャッシュエントリーへのアクセスがあったときに０にリセットされる。つまり、ニューフラグＮの１は、当該キャッシュエンドリーがリプレース（又はフィル）されてから一度もアクセスされていない、新しい状態であることを意味する。

制御部１３８は、制御部３８と比べて、設定部３７２が追加された点と、使用フラグＵおよびニューフラグＮの設定および更新を行う点とが異なる。

設定部３７２は、プロセッサ１において実行されているタスクを判別し、判別したタスクに対応するアクティブウェイデータをウェイ・レジスタ３７１に設定する。

＜設定部の構成＞

図１０は、設定部３７２の構成例を示すブロック図である。同図のように、設定部３７２は、判別部１００ａ～１００ｄとアクティブウェイデータ保持部１１０ａ～１１０ｄとセレクタ１１１とを備える。

判別部１００ａは、スタートアドレス保持部１０１、エンドアドレス保持部１０２、比較器１０３、１０４、アンド回路１０５とを有し、実行中のタスクがタスク１であるかを判別する。

スタートアドレス保持部１０１、エンドアドレス保持部１０２は、プロセッサ１から読み書き可能であり、メモリ２に格納されたタスク１のスタートアドレス、エンドアドレスをそれぞれ保持する。このスタートアドレスおよびエンドアドレスは、プロセッサ１によって予め書き込まれ、動的に変更可能である。

比較器１０３は、プロセッサ１から出力される命令フェッチアドレス（ＩＦアドレス）とスタートアドレス保持部１０１から出

力されるスタートアドレスとを比較し、スタートアドレスよりもＩＦアドレスの方が大きい場合に１を出力する。

比較器１０４は、プロセッサ１から出力されるＩＦアドレスとエンドアドレス保持部１０２から出力されるエンドアドレスとを比較し、ＩＦアドレスよりもエンドアドレスの方が大きい場合に１を出力する。

アンド回路１０５は、比較器１０３および１０４の比較結果が共に１の場合、すなわち、ＩＦアドレスがタスク１の命令をフェッチしている場合に、実行されているタスクがタスク１であることを示す。

判別部１００ｂ～１００ｄについても同様であり、実行中のタスクがタスク２～３であるかを判別する。

アクティブウェイデータ保持部１１０ａ～１１０ｄは、プロセッサ１から読み書き可能であり、判別部１００ａ～１００ｄに対応するタスクのアクティブウェイデータを保持する。このアクティブウェイデータは、プロセッサ１によって予め書き込まれ、動的に変更可能である。

セレクタ１１１は、判別部１００ａ～１００ｄの判別結果に従って、実行中のタスクに対応するアクティブウェイデータを選択し、ウェイ・レジスタ３７１に出力する。これにより、ウェイ・レジスタ３７１は、実行中のタスクに対応するアクティブウェイデータを保持する。

＜使用フラグの更新例＞

図１１は、制御部１３８による使用フラグＵの更新例を示す説明図である。同図では、説明の便宜上８ウェイではなく４ウェイの場合について説明する。同図の上段、中断、下段は、ウェイ０

～３に跨るセットＮを構成する４つのキャッシュエントリーを示している。４つのキャッシュエントリー右端の１又は０は、それぞれ使用フラグの値である。この４つの使用フラグＵをＵ０～Ｕ３と記す。

同図上段では（Ｕ０～Ｕ３）＝（１、０、１、０）であるので、ウェイ０、２のキャッシュエントリーはアクセスがあったことを、ウェイ１、３のキャッシュエントリーはアクセスがないことを意味する。

この状態で、メモリアクセスがセットＮ内のウェイ１のキャッシュエントリーにヒットした場合、同図中断に示すように、（Ｕ０～Ｕ３）＝（１、１、１、０）に更新される。つまり、実線に示すようにウェイ１の使用フラグＵ１が０から１に更新される。

さらに、同図中断の状態で、メモリアクセスがセットＮ内のウェイ３のキャッシュエントリーにヒットした場合、同図下断に示すように、（Ｕ０～Ｕ３）＝（０、０、０、１）に更新される。つまり、実線に示すようにウェイ３の使用フラグＵ１が０から１に更新される。加えて、破線に示すようにウェイ３以外の使用フラグＵ０～Ｕ２が１から０に更新される。これにより、ウェイ３のキャシュエントリーが、ウェイ０～２の各キャッシュエントリーよりも新しくアクセスされたことを意味することになる。

制御部１３８は、キャッシュミス時に使用フラグに基づいてリプレース対象のキャッシュエントリーを決定してリプレースを行う。例えば、制御部１３８は、図１１上段では、ウェイ１とウェイ３の何れかをリプレース対象と決定し、図１１中断ではウェイ３をリプレース対象と決定し、図１１下段ではウェイ０～２の何れかをリプレース対象と決定する。

＜使用フラグ、ニューフラグの更新処理＞

図１２は、制御部１３８における使用フラグおよびニューフラグのフラグ更新処理を示すフローチャートである。同図では、バリッドフラグが０（無効）であるキャッシュエントリーの使用フラグＵは０に初期化されているものとする。

同図において、制御部１３８は、キャッシュヒットしたとき（ステップＳ６１）、セットインデックスにより選択されたセットにおけるヒットしたウェイの使用フラグＵを１にセットし（ステップＳ６２）、選択されたセット内のヒットしたウェイのキャッシュエントリーのニューフラグが１なら０にリセットする（ステップＳ１７１）。

さらに、制御部１３８は、そのセット内の他のウェイの使用フラグＵを読み出し（ステップＳ６３）、読み出した使用フラグＵが全て１であるか否かを判定し（ステップＳ６４）、全て１でなければ終了し、全て１であれば他のウェイの全ての使用フラグＵを０にリセットする（ステップＳ６５）。

このようにして制御部１３８は、図１１に示した更新例のように、使用フラグを更新する。また、ニューフラグＮは、キャッシュエントリーのリプレース後、最初にアクセスされた時点でリセットされる。

＜リプレース処理＞

図１３は、制御部１３８におけるリプレース処理フローを示す図である。同図において制御部１３８は、メモリアクセスがミスしたとき（ステップＳ９１）、セットインデックスにより選択されたセットにおける、８つウェイの使用フラグＵと、８つのニューフラグＮ０～Ｎ７を読み出し（ステップＳ９２）、読み出した

８つのニューフラグＮ０～Ｎ７の全てが１であるか否かを判定し（ステップＳ１６１）、全てが１である場合は、ステップＳ９３に進み、全てが１ではない（０がある）場合には、使用フラグＵが０のウェイのうち、ニューフラグＮが１のウェイを除外する（ステップＳ１６２）。

さらに、制御部１３８は、使用フラグＵが０のウェイを１つ選択する（ステップＳ９３）。このとき、使用フラグＵが０になっているウェイが複数存在する場合は、制御部１３８はランダムに１つを選択する、あるいはラウンドロビン方式で１つを選択する。さらに、制御部１３８は、当該セットにおける選択されたウェイのキャッシュエントリーを対象にリプレースし（ステップＳ９４）、リプレース後に当該キャッシュエントリーの使用フラグＵを１に、ニューフラグを１に初期化する（ステップＳ９５）。なお、このときバリッドフラグＶ、ダーティフラグＤは、それぞれ１、０に初期化される。また、ラウンドロビン方式で、使用フラグＵが０になっている複数ウェイから１つのウェイを選択するために、制御部１３８をラウンド位置（選択したウェイの位置）を示すデータをレジスタに保持･更新し、使用フラグＵが０になっているウェイのうち次のラウンド位置を選択すればよい。

このように、リプレース対象は、ニューフラグが０でかつ使用フラグが０のキャッシュエントリーを１つ選択することにより決定される。ただし、８つのニューフラグの全てが１である場合には、ニューフラグが１でかつ使用フラグＵが０のウェイの中からリプレース対象を１つ選択する。このリプレースアルゴリズムは、従来のＬＲＵ方式におけるアクセス順序を示すデータの代わりに１ビットの使用フラグを用いるので、擬似的なＬＲＵ方式と

いうことができる。

以上説明してきたように、本実施の形態におけるキャッシュメモリによれば、設定部３７２を備えることにより、キャッシュメモリ自身が実行中のタスクを判別して、判別したタスクに対応するアクティブウェイデータをウェイ・レジスタ３７１に設定し、タスク毎にアクティブなウェイを切り替えることができる。その結果、実施の形態１と同様に、タスク切り替えに伴う無駄なリプレースの発生を低減させることができ、ヒット率を向上させることができる。

また、本実施の形態におけるキャッシュメモリによれば、従来のＬＲＵ方式におけるアクセス順序を示すデータをキャッシュエントリー毎に設ける代わりに、１ビットの使用フラグをキャッシュエントリー毎に設けている。これにより、従来のアクセス順序データを更新する複雑な回路を、使用フラグを更新する簡単なフラグ更新回路（フラグ更新部３９）に置き換えることができる。また、リプレース部４０において、リプレース対象を、使用フラグが０のキャッシュエントリーの１つを選択することにより簡単に決定することができる。このように、本実施の形態におけるキャッシュメモリによれば、ハードウェア規模を大きく低減することができる。しかも、従来のＬＲＵと比較してもほぼ同等のヒット率を得ることができる。

さらに、本実施の形態における制御部１３８は、ニューフラグが１の場合は、当該キャッシュエントリーをリプレース対象から除外している。これは、次の理由による。すなわち、使用フラグＵは初期値が１であるが他のウェイの使用フラグが順次１になれば、０にリセットされる。つまり、使用フラグＵが０のキャッ

シュエントリーであってもリプレース後に一度もアクセスされていない場合がある。こうして使用フラグが０になった場合、リプレース後に一度もアクセスされていないキャッシュエントリーが、キャッシュミスの発生により再度リプレース対象に選択されてしまう可能性がある。そのため、ニューフラグＮを設けることにより、リプレースされた後に一度もアクセスされていないキャッシュエントリーがリプレースされてしまうことを防止することができる。

＜変形例＞

なお、本発明のキャッシュメモリは、上記の実施の形態の構成に限るものではなく、種々の変形が可能である。以下、変形例のいくつかについて説明する。

（１）実施の形態１における変形例（１）～（６）を本実施の形態に適用してもよい。

（２）プロセッサ１から実行中のタスクを示すタスク番号（あるいはスレッド番号、プロセス番号等）が出力される場合には、上記判別部１００ａ～１００ｄの代わりに、タスク番号を保持および更新するタスク番号保持部を備える構成としてもよい。この場合セレクタ１１１は、タスク番号に対応するアクティブウェイデータを選択すればよい。

（３）制御部１３８は、図１１の下段に示したようにセット内の他のウェイの使用フラグＵが全部１であれば０にし、ヒットしたウェイ自身の使用フラグＵを１に更新するが、この代わりに、ヒットしたウェイ自身の使用フラグも０に更新する構成としてもよい。

（４）ウェイ・レジスタ３７１の内容が更新されたとき、制御部

１３８は、全ての使用フラグをリセットする構成としてもよい。さらに、使用フラグのリセットと共に、制御部１３８は、使用フラグＵが０になっている複数のウェイから１つを選択するための上記ラウンドロビン方式におけるラウンド位置を示す情報をリセットしてもよい。

（５）上記実施形態におけるニューフラグを有しない構成としてもよい。

（実施の形態３）

実施の形態１、２では、キャッシュエントリーのリプレース単位がライン（１２８バイト）単位でなされる構成を開示したが、本実施の形態では、リプレース単位がタスク毎にライン単位とサブライン（３２バイト）単位とで切り替え可能な構成について説明する。

図１４は、本発明の実施の形態３におけるキャッシュメモリの構成を示すブロック図である。同図のキャッシュメモリは、図８に示した構成と比較して、ウェイ１３１ａ～１３１ｈの代わりにウェイ２３１ａ～２３１ｈを備える点と、セレクタ２３３ａ～２３３ｈが追加された点と、制御部１３８の代わりに制御部２３８を備える点とが異なっている。以下、同じ点は説明を省略して異なる点を中心に説明する。

ウェイ２３１ａ～２３１ｈは、図８のウェイ１３１ａ～１３１ｈと比べて、キャッシュエントリー内にバリッドフラグとダーティフラグとを１ビットずる保持するのではなく、サブライン毎に保持する点が異なっている。図１５に、キャッシュエントリーのビット構成を示す。同図のように、同図のように１つのキャッシュエントリーは、バリッドフラグＶ０～Ｖ３、タグ、ラインデー

タ、使用フラグＵ、ニューフラグＮ、ダーティフラグＤ０～Ｄ３を保持する。使用フラグＵおよびニューフラグについては既に説明したので省略する。ラインデータ（１２８バイト）は４つのサブライン（３２バイト）からなる。バリッドフラグＶ０～Ｖ３は、４つのサブライン０～３に対応し、対応するサブラインが有効か否かを示す。ダーティフラグＤ０～Ｄ３は、４つのサブライン０～３に対応し、対応するサブラインに書き込みがあったか否かを示す。バリッドフラグおよびダーティフラグがサブライン単位に設けられているのは、リプレースをサブライン単位でも行うことを可能にするためである。また、ライトバック（又はライトスルー）もサブライン単位で行うことが可能である。

セレクタ２３３ａは、ウェイ２３１ａから、ソースインデックスＳＩにより選択されたセットに対応するバリッドフラグＶ０～Ｖ３と、ワードインデックスＷＩの上位２ビットとが入力され、この上位２ビットに指定されるサブラインに対応するバリッドフラグを選択する。セレクタ２３３ｂ～２３３ｈについても、ウェイ２３１ｂ～２３１ｈに対応している点以外同様である。これによりセレクタ２３３ａ～２３３ｈは、サブライン単位でヒットしたか否かを判定すること可能にしている。

制御部２３８は、制御部１３８と比べて、設定部３７２が削除された点と、ＲＳ（リプレースサイズ）レジスタ３７３が追加された点とが異なる。

設定部３７２が削除されているのは、ウェイ・レジスタ３７１が実施の形態１と同様にタスク切り替えにおいて書き換えられるからである。

ＲＳレジスタ３７３は、ウェイ毎にリプレースサイズを示すリ

プレースサイズデータを保持する。図１６にＲＳレジスタ３７３のビット構成例を示す。同図のようにＲＳレジスタ３７３は、ＲＳ０～ＲＳ７からなるリプレースサイズデータを保持する。ＲＳ０～ＲＳ７の各ビットは、１のときリプレースサイズがサブライン（３２バイト）であることを、０のときリプレースサイズがライン（１２８バイト）であることを制御部２３８に指示する。このＲＳレジスタ３７３は、ウェイ・レジスタ３７１と同様に、プロセッサ１から読み書き可能であり、コンテキストの一部としてタスク切り換えにおいて書き換えられる。これにより、リプレースサイズをラインとするとサブラインとするかを、ウェイ毎にかつ
タスク毎に設定することを可能にしている。

図１７は、制御部２３８におけるフラグの更新処理を示すフローチャートである。同図は、図１２に示したフローチャートと比べて、ダーティフラグをサブライン単位で設定するためのステップＳ１７５～Ｓ１７７が追加された点が異なる。すなわち、制御部２３８は、キャッシュメモリへの書き込みがあったとき（ステップＳ１７５）、書き込まれたサブラインを判別し（ステップＳ１７６）、判別されたサブラインに対応するダーティフラグを１にセットする（ステップＳ１７７）。ステップＳ１７５～Ｓ１７７の処理は、例えば、制御部２３８に１入力４出力のデマルチプレクサをウェイ毎に備えることにより簡単に実現することができる。このデマルチプレクサは、論理”１”が入力され、４つの出力をキャッシュエントリー中のダーティフラグＤ０～Ｄ３に対応させ、ワードインデックスＷＩの上位２ビットにより出力先を制御するよう構成すればよい。

このようにして、制御部２３８は、サブライン単位に設けられたダーティフラグＤ０～Ｄ３をキャッシュライトに応じて更新する。

図１８は、制御部２３８におけるリプレース処理を示すフローチャートである。同図は、図１３に示したフローチャートと比べて、ステップＳ９４の代わりにステップＳ１８１～１８３を有する点と、ステップＳ９５の代わりにステップＳ９５ａを有する点とが異なる。

制御部２３８は、ＲＳレジスタ３７３からステップＳ９３で選択されたウェイに対応するＲＳフラグを読み出して、リプレースサイズとしてサブラインとラインの何れが指定されているかを判定し（ステップＳ１８１）、サブラインと指定されている場合には、当該ウェイのサブラインをリプレースし（ステップＳ１８２）、ラインと指定されている場合には、当該ウェイのラインをリプレースする（ステップＳ１８２）。さらに、制御部２３８は、リプレースされたサブラインまたはラインに対応するバリッドフラグおよびダーティフラグを初期化する（ステップＳ９５ａ）。すなわち、リプレースされたサブラインに対応するバリッドフラグ、ダーティフラグをそれぞれ１、０に設定する。ライン単位でリプレースされた場合には、４つのサブラインに対応する４つのバリッドフラグ、ダーティフラグをそれぞれ１、０に設定する。

以上説明してきたように本実施の形態におけるキャッシュメモリによれば、実施の形態１又は２に加えて、リプレース単位をラインとサブラインとでウェイ毎およびタスク毎に設定できるので、タスクの必要とするデータサイズに応じてリプレース単位を切り換えることにより、キャッシュミスをさらに低減すること

ができる。例えば、タスク１はオーディオデータのデコード／エンコード処理を、タスク２がビデオデータのデコード／エンコード処理を行うものとする。この場合、タスク１ではラインサイズをリプレース単位とし、タスク２ではサブラインをリプレース単位とすることができる。こうすれば、タスク１、２のキャッシュ利用効率を向上させることができる。なぜなら、タスク１は、シーケンシャルアクセスするデータの長さが比較的長く、タスク２は、シーケンシャルアクセスするデータの長さが比較的短いからである。

＜変形例＞

なお、本発明のキャッシュメモリは、上記の実施の形態の構成に限るものではなく、種々の変形が可能である。以下、変形例のいくつかについて説明する。

（１）実施の形態２における変形例（１）、（３）、（４）、（５）を本実施の形態に適用してもよい。

（２）上記実施の形態では、サブラインのサイズをラインのサイズの１／４としているが、１／２、１／８、１／１６等他のサイズでもよい。その場合、各キャッシュエントリー
は、サブラインと同数のバリッドフラグおよびダーティフラグをそれぞれ保持すればよい。

（実施の形態４）

実施の形態１～３では、インアクティブなウェイに対して少なくともリプレースが制限される例を説明したが、本実施の形態では、さらに比較器３２ａ～３２ｈのうちインアクティブなウェイに対応する比較器における比較の禁止と、インアクティブなウェイに対応するキャッシュエントリーからのタグ出力の禁止をす

る構成について説明する。

そのため本実施の形態におけるキャッシュメモリは図１４に示した制御部２３８内部に比較制御部３７２を追加した構成となっている。

図１９は、比較制御部３７２およびウェイ２３１ａ～２３１ｈの要部の構成を示すブロック図である。

同図において、キャッシュアドレスエントリ３００ａ～３００ｈは、キャッシュメモリ中のウェイ０～８に含まれ、それぞれセット数と同数（実施形態では１６個）のタグを保持する。キャッシュアドレスエントリ３００ａは、ウェイ０に含まれ、１６個のタグ保持部３０１～３１６と、１６個のアンド回路３２１～３３６とを含む。他のキャッシュアドレスエントリ３００ｂ～３００ｈも同様である。

アンド回路３２１は、１６個のセット中のセット０に対応し、セットインデックスをデコードするデコーダ３０によりセット０が選択信号（ｓｅｔ０）と、イネーブル信号Ｅ０とのアンドをとる。その結果、アンド回路３２１は、選択信号ｓｅｔ０＝１でかつイネーブル信号Ｅ０＝１のときのみ、タグ保持部３０１からのタグ出力と、比較器３２ａとをイネーブルにする。

比較器３２ａ～３２ｈは、それぞれイネーブル端子ＥＮを有し、イネーブル端子の入力が１のときに、アドレスレジスタ２０中のタグアドレスとキャッシュアドレスエントリ３０１からのタグとの比較動作を行う。

比較制御部３７２は、８つのウェイに対応する８つのイネーブル回路３８１ａ～３８１ｈ、回数カウンタ３８２を備え、比較器３２ａ～３２ｈにおいて最大２回のタグ比較を行わせるよう制

御し、１回目のタグ比較では、ウェイ・レジスタ３７１に示されたアクティブなウェイに対応する比較器をイネーブルにし、かつインアクティブウェイに対応する比較器をディスエーブルすることによって、アクティブウェイのタグを比較対象とし、さらに、１回目のタグ比較においてミスヒットと判定された場合に、アクティブウェイに対応する比較器をディスエーブルし、アクティブウェイに対応する比較器をイネーブルにすることによって、インアクティブウェイのタグを比較対象とし２回目のタグ比較を行わせるよう構成している。また、各比較器のイネーブル／ディスエーブルの制御と同時にウェイ２３１ａ～２３１ｈからのタグ出力もイネーブル／ディスエーブルを制御している。これによりディスエーブルされた比較器およびタグ出力による消費電力の低減を図っている。

イネーブル回路３８１ａ～３８１ｈは、ウェイ２３１ａ～２３１ｈに対応し、ウェイ・レジスタ３７１に保持されるアクティブウェイデータに従って、比較器３２ａ～３２ｈのうち、１回目のタグ比較ではアクティブなウェイに対応する比較器のみをイネーブルにし、２回目のタグ比較ではインアクティブなウェイに対応する比較器のみをイネーブルにする。

すなわち、イネーブル回路３８１ａ～３８１ｈは、イネーブル信号Ｅ０～Ｅ７を生成し、このイネーブル信号Ｅ０～Ｅ７により、キャッシュメモリ中のウェイ０～８に対応する８つのキャッシュアドレスエントリーからのタグ出力と、８つの比較器３２ａ～３２ｈをイネーブル／ディスエーブルする。例えば、イネーブル回路３８１ａは排他的論理和回路により、ウェイ０がアクティブか否かを示すＷ０ビットと、回数カウンタ３８２のカウント

値に従ってイネーブル信号Ｅ０を生成する。

回数カウンタ３８２は、比較の回数をカウントするカウンタであり、０（１回目）、１（２回目）とカウントアップする。ただし、１回目がヒットした場合にはカウントアップしない。

図２０は、回数カウンタ３８２のカウント値とアクティブウェイデータＷｎ（ｎ＝０～７）を入力として、イネーブル信号Ｅｎを出力とするイネーブル回路の制御論理を示す真理値表を示す。同図において、例えば、ウェイ０～２がアクティブでウェイ３～７がインアクティブである場合（Ｗ０＝１）、１回目の比較では、イネーブル信号Ｅ０～Ｅ２が１（イネーブル）でイネーブル信号Ｅ３～Ｅ７がディスエーブルになる。この１回目の比較でヒットした場合には、回数カウンタがカウントアップしないので２回目の比較はなされない。１回目の比較でミスヒットした場合には、回数カウンタがカウントアップするので２回目の比較がなされる。この場合、２回目の比較では、イネーブル信号Ｅ０～Ｅ２が０（ディスエーブル）でイネーブル信号Ｅ３～Ｅ７が１（イネーブル）になる。

以上の構成により、１回目のタグ比較では、インアクティブなウェイを比較対照としないので比較器における消費電力およびタグ出力による消費電力を低減することができる。さらに、１回目のタグ比較においてミスヒットした場合には、インアクティブなウェイのみを比較対象として２回目のタグ比較を行うので、全てのウェイのキャッシュデータを有効に活用することができる。

なお、２回目の比較において全ての比較器をイネーブルにしてもよい。この場合、１回目でのヒットした場合に２回目の比較が行われないので、消費電力を低減することができる。つまり、タ

スクごとにアクティブウェイが割り当てられるので１回目の比較において高いヒット率を得られると考えられる。

また、２回目の比較を行わない構成としてもよい。この１回目の比較において高いヒット率を得られると考えられるので、消費電力を低減する効果がある。

産業上の利用可能性

本発明は、メモリアクセスを高速化するためのキャッシュメモリおよびその制御方法に適しており、例えば、オンチップキャッシュメモリ、オフチップキャッシュメモリ、データキャッシュメモリ、命令キャッシュメモリ等に適している。

請　求　の　範　囲

１．Ｎ－ウェイ・セット・アソシエイティブ方式のキャッシュメモリであって、

Ｎ個のウェイのうち１つ以上のウェイを示す制御レジスタと、

制御レジスタに示されるウェイをアクティブにする制御手段と、

制御レジスタの内容を更新する更新手段と

を備えることを特徴とするキャッシュメモリ。

２．前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイに対して少なくともリプレースを制限する

ことを特徴とする請求項１記載のキャッシュメモリ。

３．前記キャッシュメモリは、さらに、

ウェイ毎に設けられ、キャッシュデータのアドレスをタグとして保持するタグ保持手段と、

プロセッサから出力されるメモリアクセスアドレスの上位部分であるタグアドレスと、タグ保持手段から出力されるＮ個のタグとを比較することによりヒットかミスヒットかを判定するＮ個の比較手段を有し、

前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイに対応する比較手段をディスエーブルにする

ことを特徴とする請求項１記載のキャッシュメモリ。

４．前記制御手段は、さらに、制御レジスタに示されたアクティ

ブなウェイ以外のウェイに対応するキャッシュアドレス保持手段に対して、比較手段へのタグ出力をディスエーブルにする

ことを特徴とする請求項３記載のキャッシュメモリ。

５．前記制御手段は、プロセッサからメモリアクセスアドレスが出力されたとき、当該アクセスアドレスについて、比較手段に最大２回のタグ比較を行わせるよう制御し、

１回目のタグ比較において制御レジスタに示されたアクティブなウェイ以外のウェイに対応する比較手段をディスエーブルし、

１回目のタグ比較においてミスヒットと判定された場合に、アクティブなウェイ以外のウェイに対応する比較手段をディスエーブルしないで２回目のタグ比較を行わせる

ことを特徴とする請求項３記載のキャッシュメモリ。

６．前記制御手段は、前記２回目のタグ比較においてアクティブなウェイに対応する比較手段をディスエーブルする

ことを特徴とする請求項５記載のキャッシュメモリ。

７．前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイに対して、その状態の更新を禁止する

ことを特徴とする請求項２記載のキャッシュメモリ。

８．前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイについて、そのアクセス順序を示す情報の更新を禁止する

ことを特徴とする請求項２記載のキャッシュメモリ。

９．前記キャッシュメモリは、さらに

前記更新手段によって制御レジスタの内容が更新されたとき、ウェイに対するアクセス順序を示す情報をリセットするリセット手段を有する

ことを特徴とする請求項２記載のキャッシュメモリ。

１０．前記アクセス順序を示す情報は、キャッシュエントリー毎の１ビットデータであり、

前記キャッシュメモリは、さらに、リプレース可能な複数ウェイから１つのウェイをラウンドロビン方式で選択するためラウンド位置を示すデータを保持するレジスタを有し、

前記リセット手段は、前記更新手段によって制御レジスタの内容が更新されたとき、前記レジスタをリセットする

ことを特徴とする請求項９記載のキャッシュメモリ。

１１．前記更新手段は、

アクティブにすべきウェイを指定するウェイデータであって、タスク毎のウェイデータを保持する保持手段と、

実行中のタスクに対応するウェイデータを保持するよう前記制御レジスタを書き換える書き換え手段と

を有することを特徴とする請求項２記載のキャッシュメモリ。

１２．前記保持手段は、メモリ中に記憶されたタスク毎のコンテキストデータの一部として前記ウェイデータを保持し、

前記書き換え手段は、タスク切り替えに際して、制御レジスタ中の現タスクのウェイデータをメモリに退避し、次タスクのウェイデータをメモリから前記制御レジスタに復帰する

ことを特徴とする請求項１１記載のキャッシュメモリ。

１３．前記保持手段は、タスク毎の前記ウェイデータを保持し、

前記書き換え手段は、

メモリに記憶された各タスクのアドレス範囲を記憶するアドレス記憶手段と、

アドレス記憶手段に記憶されたアドレス範囲と、プロセッサから出力される命令フェッチアドレスとに基づいて、実行中のタスクを判別する判別手段と、

判別された実行中のタスクに対応するウェイデータを前記保持手段から選択する選択手段と、

選択されたウェイデータを前記制御レジスタに書き込む書き込み手段と

を備えることを特徴とする請求項１２記載のキャッシュメモリ。

１４．前記保持手段は、タスク毎の前記ウェイデータを保持し、

前記書き換え手段は、

プロセッサから出力されるタスク番号に従って、実行中のタスクに対応するウェイデータを前記保持手段から選択する選択手段と、

選択されたウェイデータを前記制御レジスタに書き込む書き込み手段と

を備えることを特徴とする請求項１２記載のキャッシュメモリ。

１５．前記保持手段に保持されるウェイデータは、オペレーティングシステムによってタスクに割り当てられる
ことを特徴とする請求項１１記載のキャッシュメモリ。

１６．前記キャッシュメモリは、各ウェイにおけるリプレース単位をキャッシュエントリーのラインサイズと、ラインサイズの２のｎ乗分の１のサイズとに切り替え可能であり、
前記制御レジスタは、さらに、ウェイ毎のリプレースサイズを示し、
前記制御手段は、制御レジスタに示されたリプレースサイズを単位としてリプレース制御を行う
ことを特徴とする請求項１記載のキャッシュメモリ。

１７．前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイに対して少なくともリプレースを制限し、制御レジスタに示されたアクティブなウェイに対して制御レジスタに示されたサイズを単位にリプレースを行う
ことを特徴とする請求項１６記載のキャッシュメモリ。

１８．前記更新手段は、
アクティブにすべきウェイを指定するウェイデータであってタスク毎のウェイデータと、タスク毎のリプレースサイズとを保持する保持手段と、

実行中のタスクに対応するウェイデータ及びリプレースサイズを保持するよう前記制御レジスタを書き換える書き換え手段と

を有することを特徴とする請求項１７記載のキャッシュメモリ。

１９．前記キャッシュメモリは、さらに、

キャッシュの単位となるデータを保持するキャッシュエントリー毎に、アクセスの有無を示す１ビットのアクセス情報を記憶する記憶手段と、

アクセス無しを示すアクセス情報に対応するキャッシュエントリーの中からリプレース対象のキャッシュエントリーを選択する選択手段と

を備えることを特徴とする請求項１記載のキャッシュメモリ。

２０．前記キャッシュメモリは、さらに

リプレース可能な複数ウェイから１つのウェイをラウンドロビン方式で選択するためのラウンド位置を示すデータを保持するレジスタを有し、

前記更新手段によって制御レジスタの内容が更新されたとき、ウェイに対するアクセス順序を示す情報と、前記レジスタをラウンド位置を示すデータとをリセットするリセット手段と

を有することを特徴とする請求項１９記載のキャッシュメモリ。

２１．Ｎ－ウェイ・セット・アソシエイティブ方式のキャッシュメモリを制御する制御方法であって、

Ｎ個のウェイのうち１つ以上のウェイを示すウェイデータを制御レジスタに設定するステップと、

制御レジスタに示されるウェイをアクティブにする制御ステップと

を有することを特徴とする制御方法。

２２．前記制御ステップでは、制御レジスタに示されたアクティブなウェイ以外のウェイに対して少なくともリプレースを制限する

ことを特徴とする請求項２１記載の制御方法。

２３．前記制御方法は、さらに、

アクティブにすべきウェイを指定するウェイデータであってタスク毎のウェイデータを保持する保持部から、実行中のタスクに対応するウェイデータ読み出して、読み出したウェイデータを前記制御レジスタに書き込む更新ステップを有する

ことを特徴とする請求項２２記載の制御方法。

要　約　書

本発明のキャッシュメモリは、Ｎ－ウェイ・セット・アソシエイティブ方式のキャッシュメモリであって、Ｎ個のウェイのうち１つ以上のウェイを示す制御レジスタと、制御レジスタに示されるウェイをアクティブにする制御手段と、制御レジスタの内容を更新する更新手段とを備え、前記制御手段は、制御レジスタに示されたアクティブなウェイ以外のウェイに対して少なくともリプレースを制限する。

図1

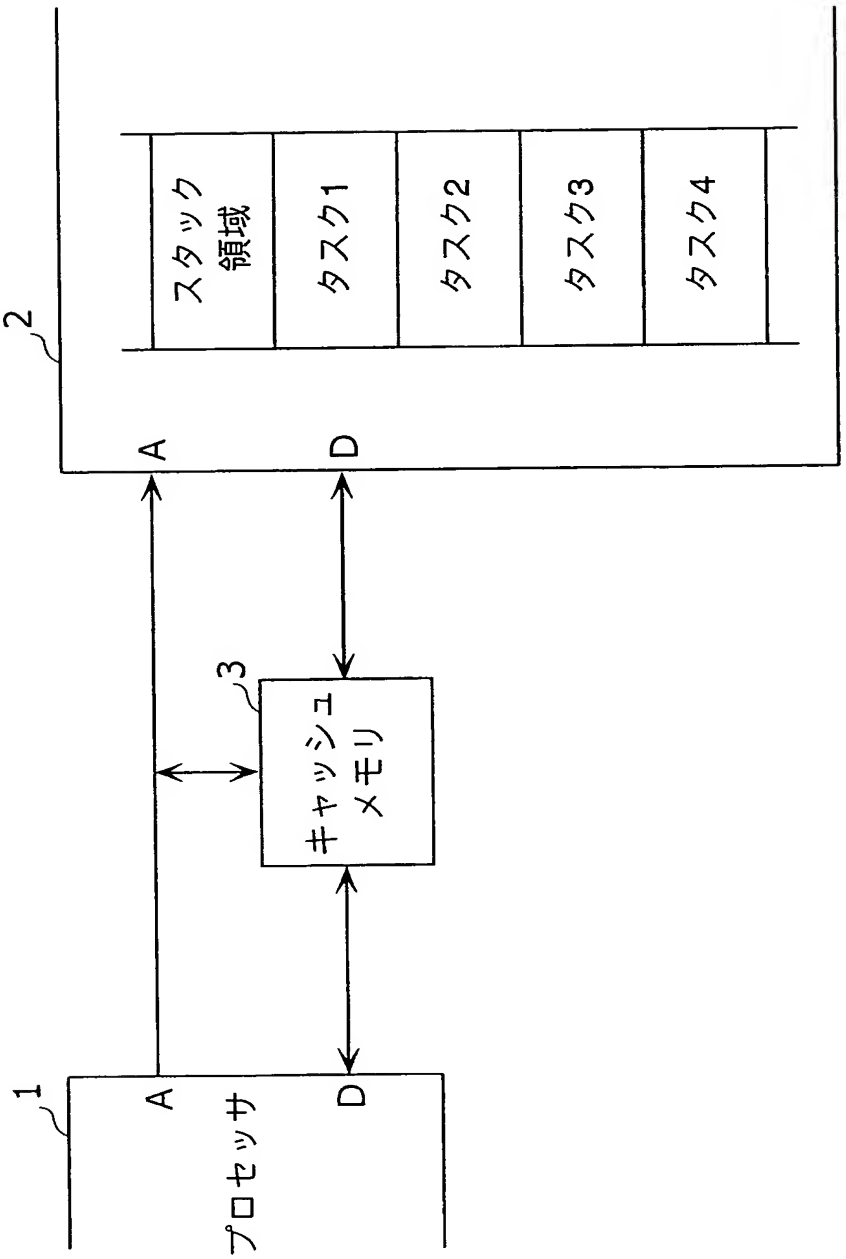


図2

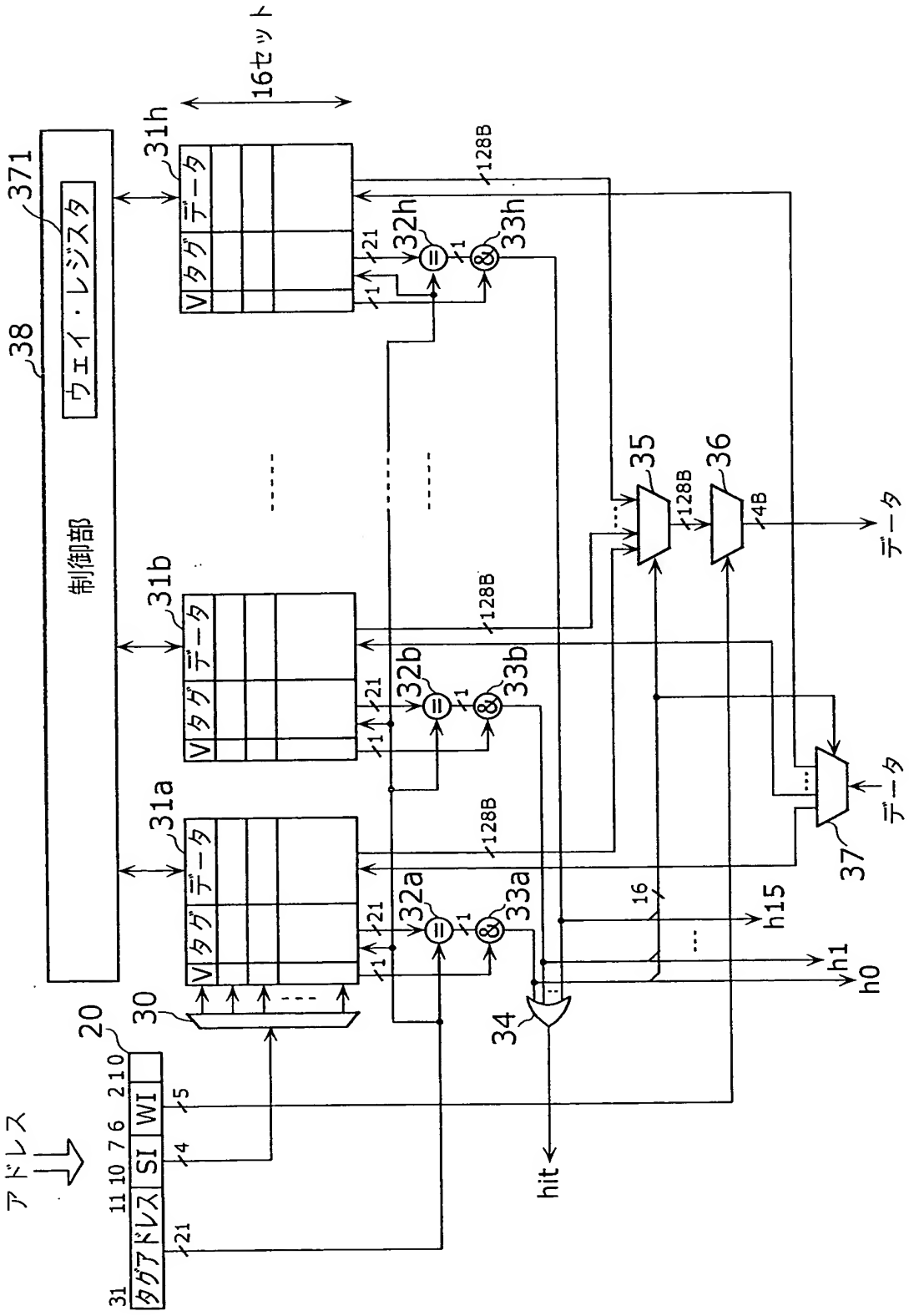

16セット
371
ウェイ・レジスタ
38
制御部
31h
31b
31a
V
タグ
データ
128B
21
1
32h
32b
32a
33h
33b
33a
35
36
37
4B
データ
16
h15
h1
h0
hit
34
30
20
アドレス
31
タグアドレス
11 10 7 6 2 1 0
SI
WI
5
4
21

図3

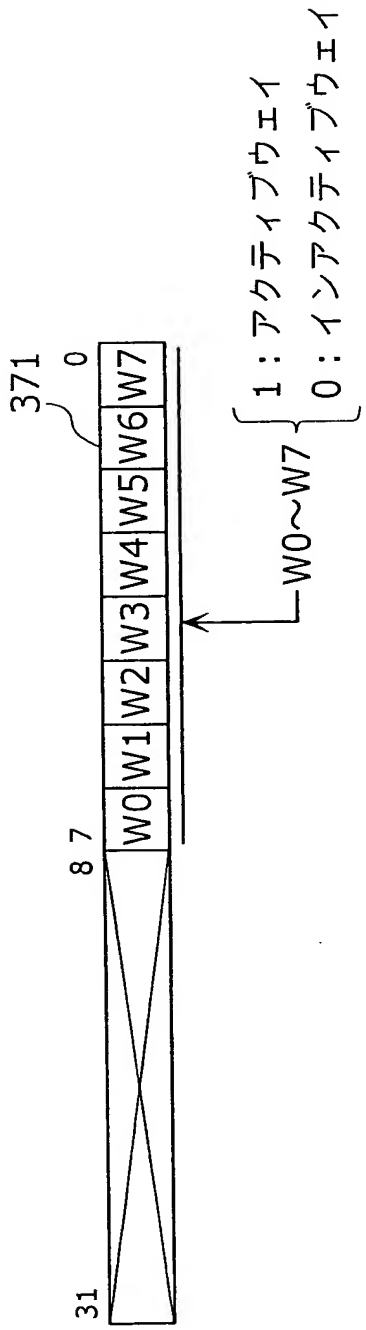


図4

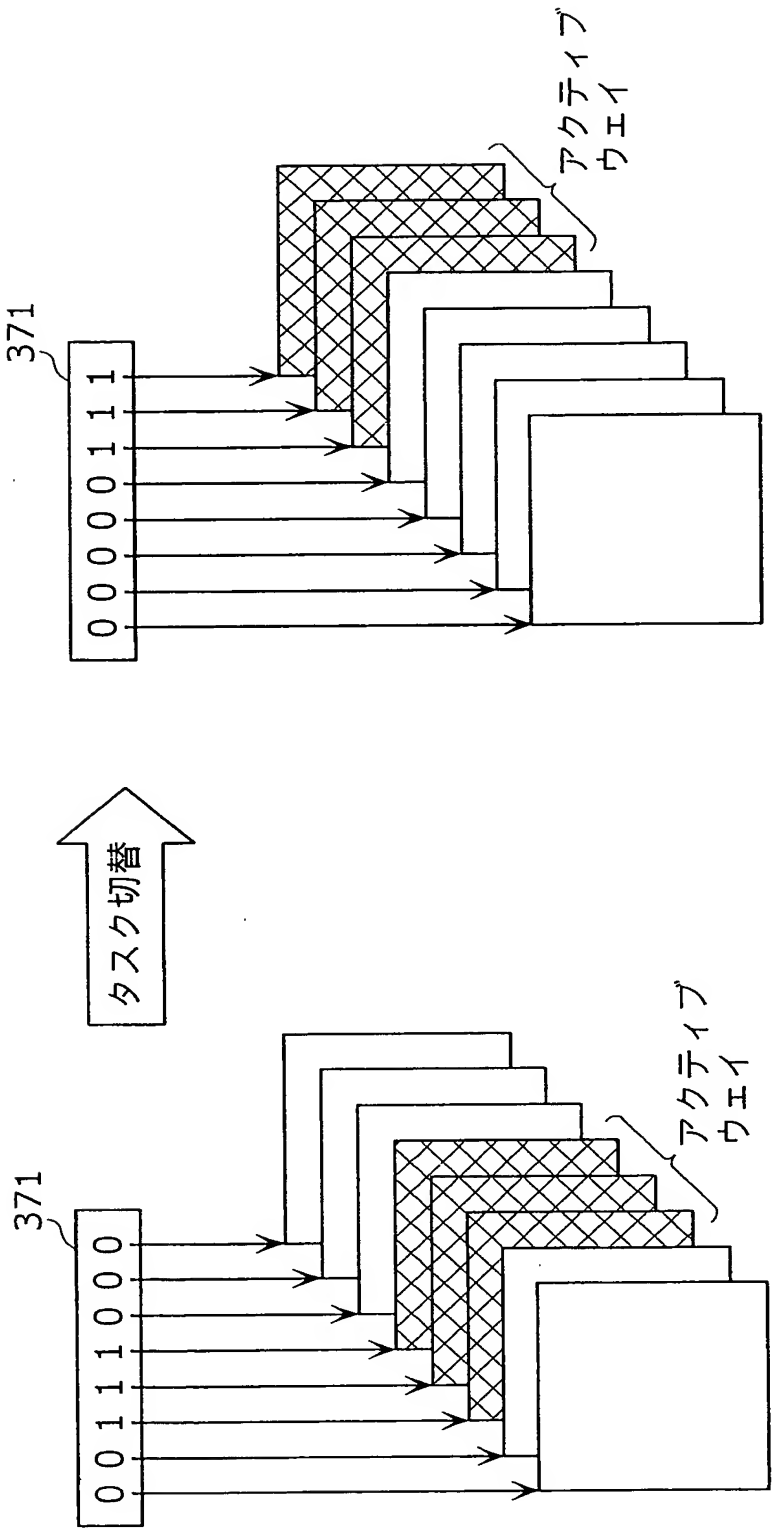


図5

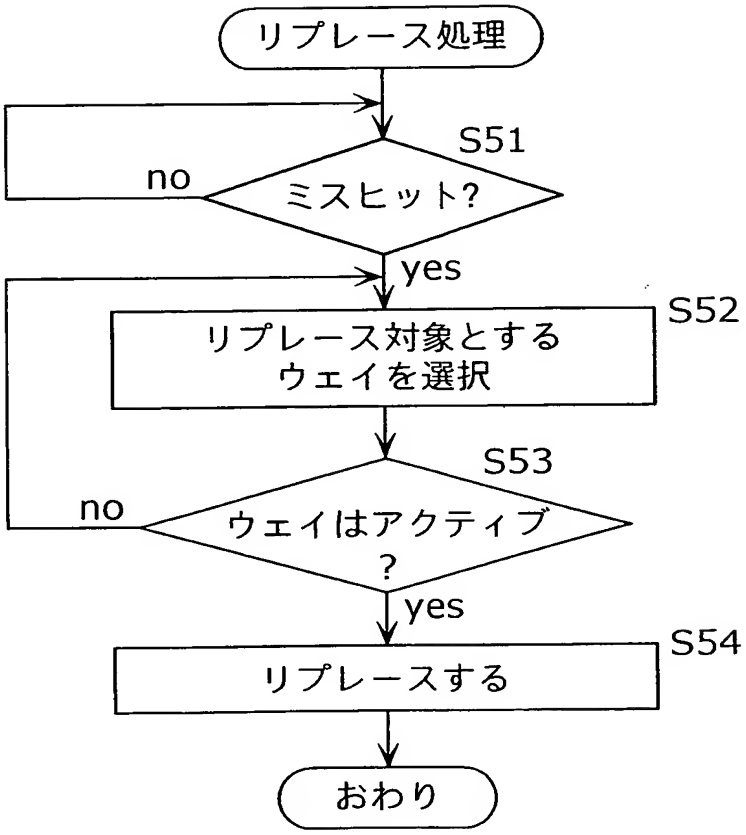

リプレース処理
S51
no
ミスヒット?
yes
S52
リプレース対象とする
ウェイを選択
S53
no
ウェイはアクティブ
?
yes
S54
リプレースする
おわり

図6

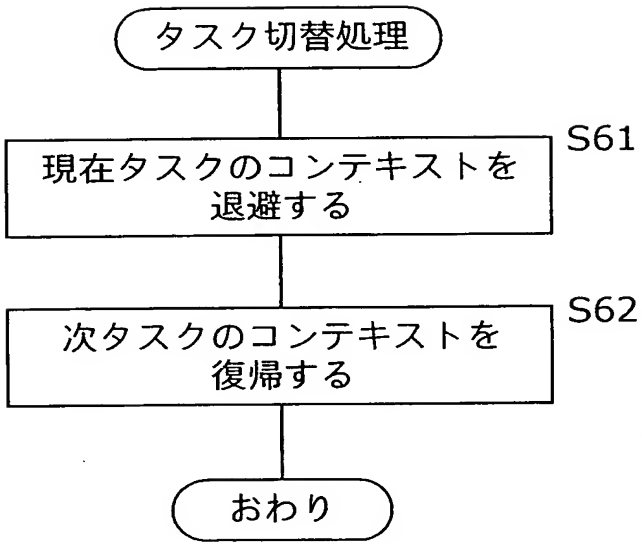
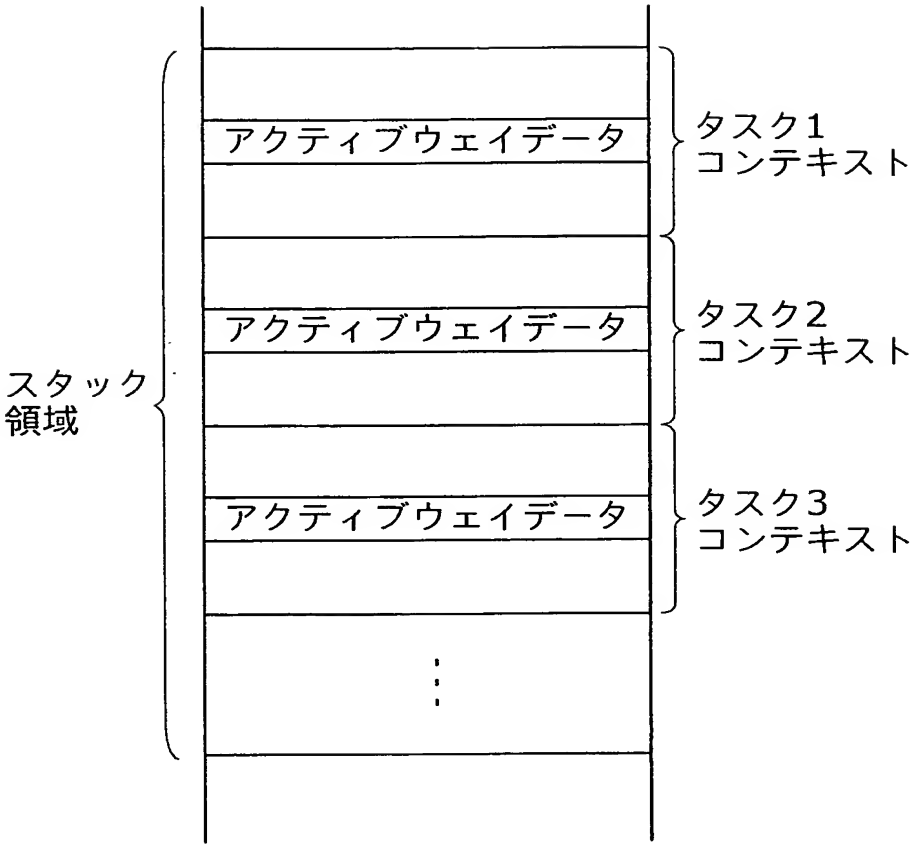


図7


スタック
領域
アクティブウェイデータ
アクティブウェイデータ
アクティブウェイデータ
タスク1
コンテキスト
タスク2
コンテキスト
タスク3
コンテキスト

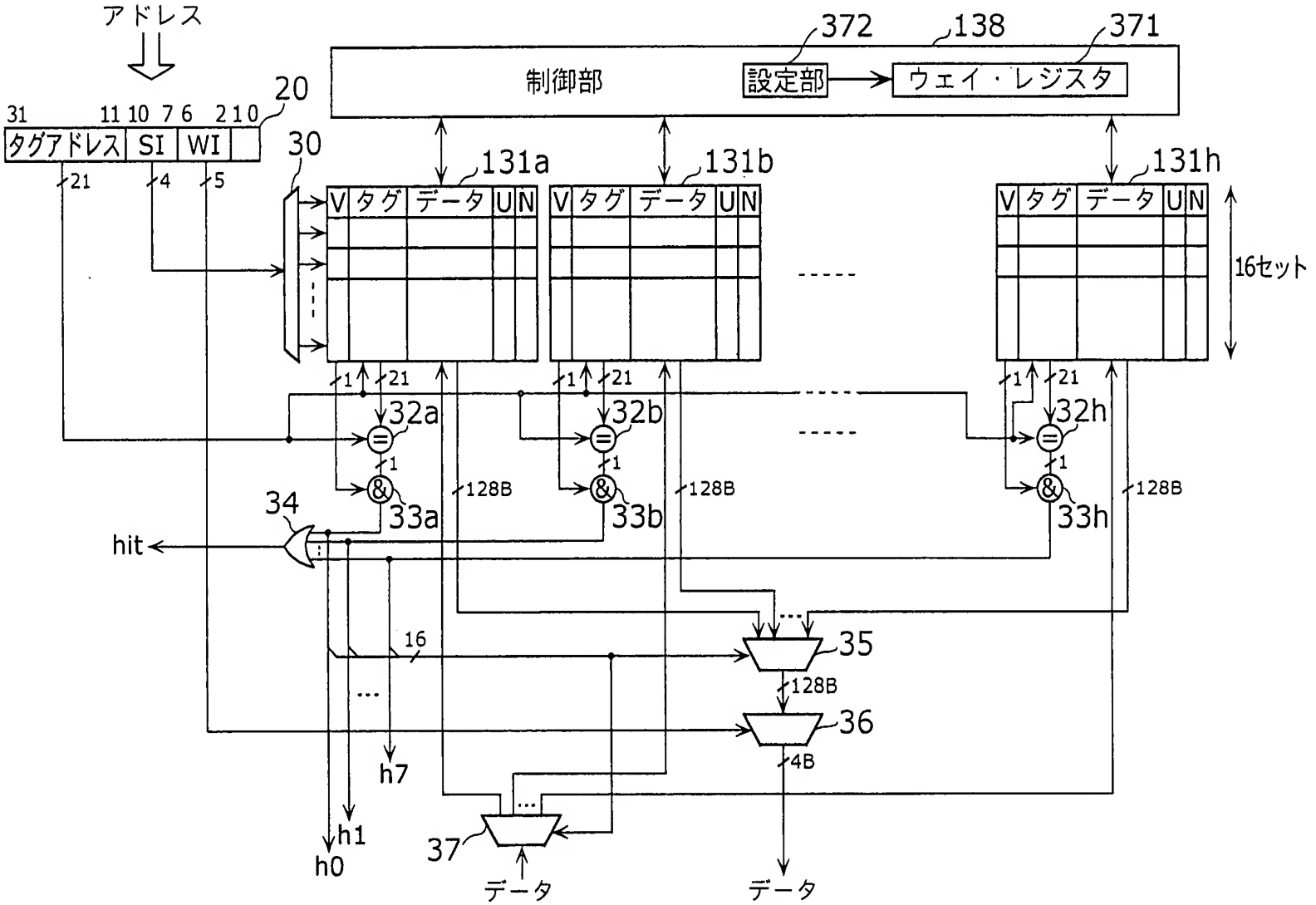

アドレス
372
138
371
制御部
設定部
ウェイ・レジスタ
20
31
11 10 7 6 2 1 0
タグアドレス
SI
WI
21
4
5
30
131a
131b
131h
V タグ データ U N
16セット
1
21
32a
32b
32h
33a
33b
33h
128B
34
hit
16
35
36
4B
h7
h1
h0
37
データ
データ

図8

図9

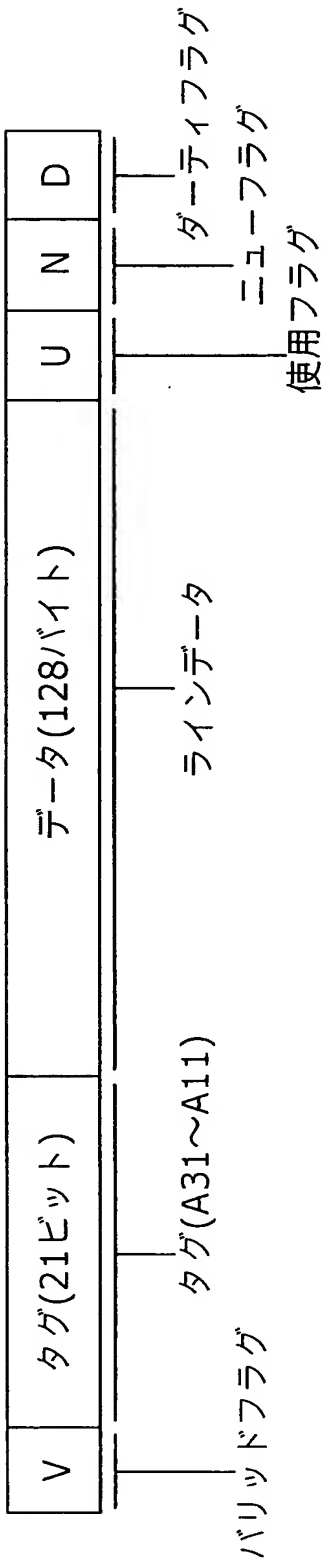

V
タグ(21ビット)
データ(128バイト)
U
N
D
バリッドフラグ
タグ(A31～A11)
ラインデータ
使用フラグ
ニューフラグ
ダーティフラグ

図10

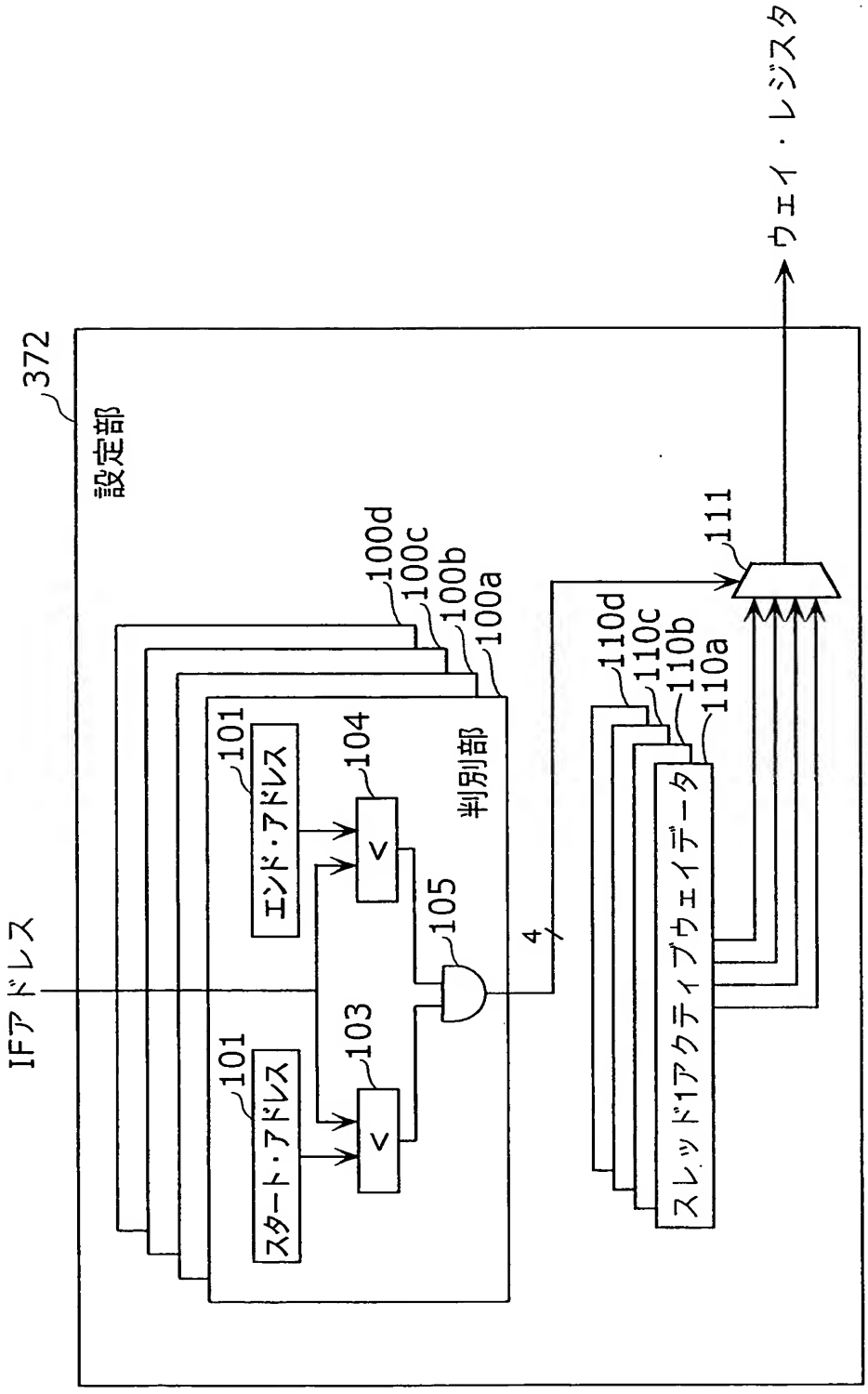

372
設定部
IFアドレス
101
スタート・アドレス
101
エンド・アドレス
103
<
104
<
105
判別部
100a
100b
100c
100d
4
111
110a
110b
110c
110d
スレッド1アクティブウェイデータ
ウェイ・レジスタ

図11

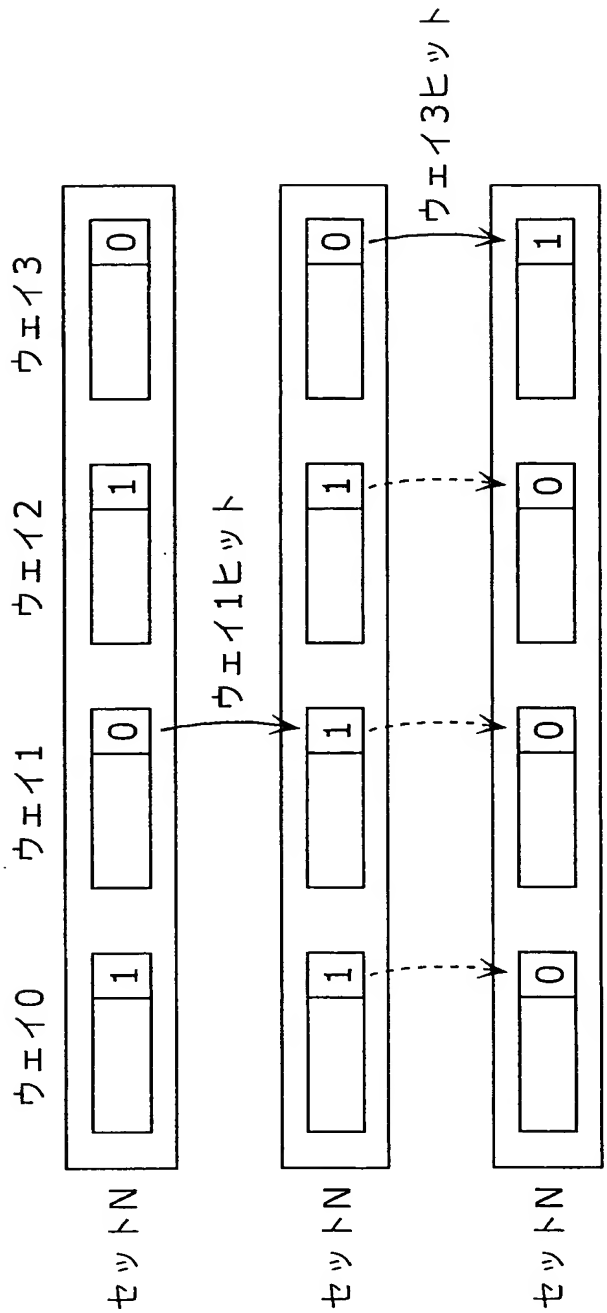

ウェイ0
ウェイ1
ウェイ2
ウェイ3
セットN
セットN
セットN
ウェイ1ヒット
ウェイ3ヒット
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0

図12

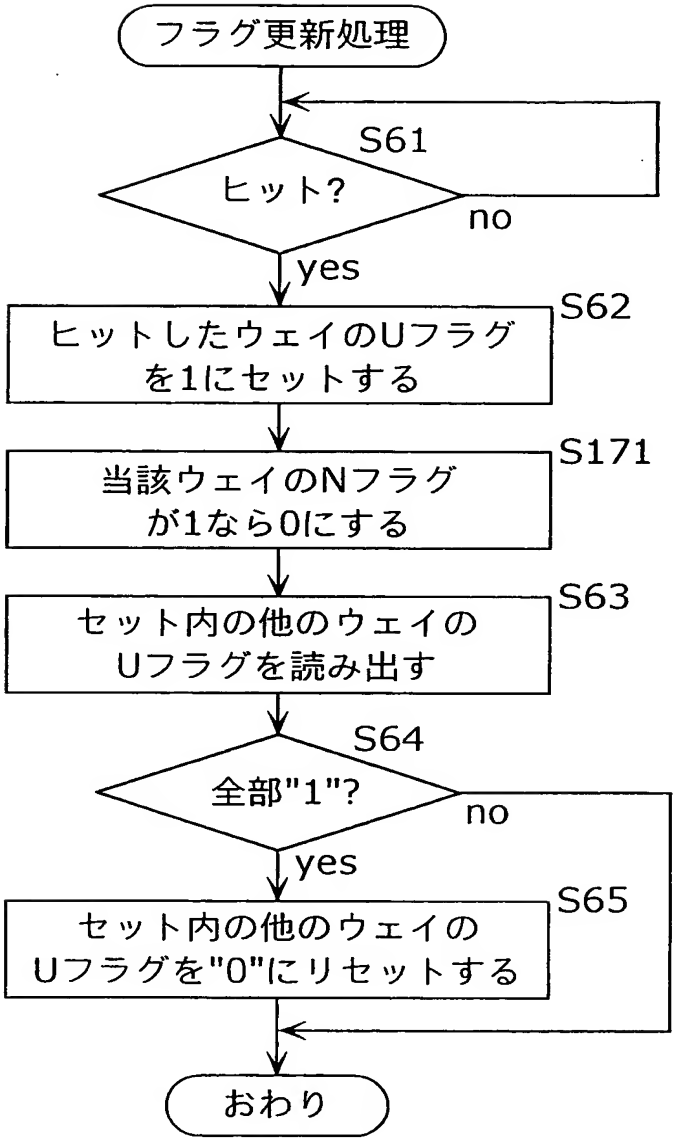

フラグ更新処理
S61
ヒット?
no
yes
S62
ヒットしたウェイのUフラグ
を1にセットする
S171
当該ウェイのNフラグ
が1なら0にする
S63
セット内の他のウェイの
Uフラグを読み出す
S64
全部"1"?
no
yes
S65
セット内の他のウェイの
Uフラグを"0"にリセットする
おわり

図13

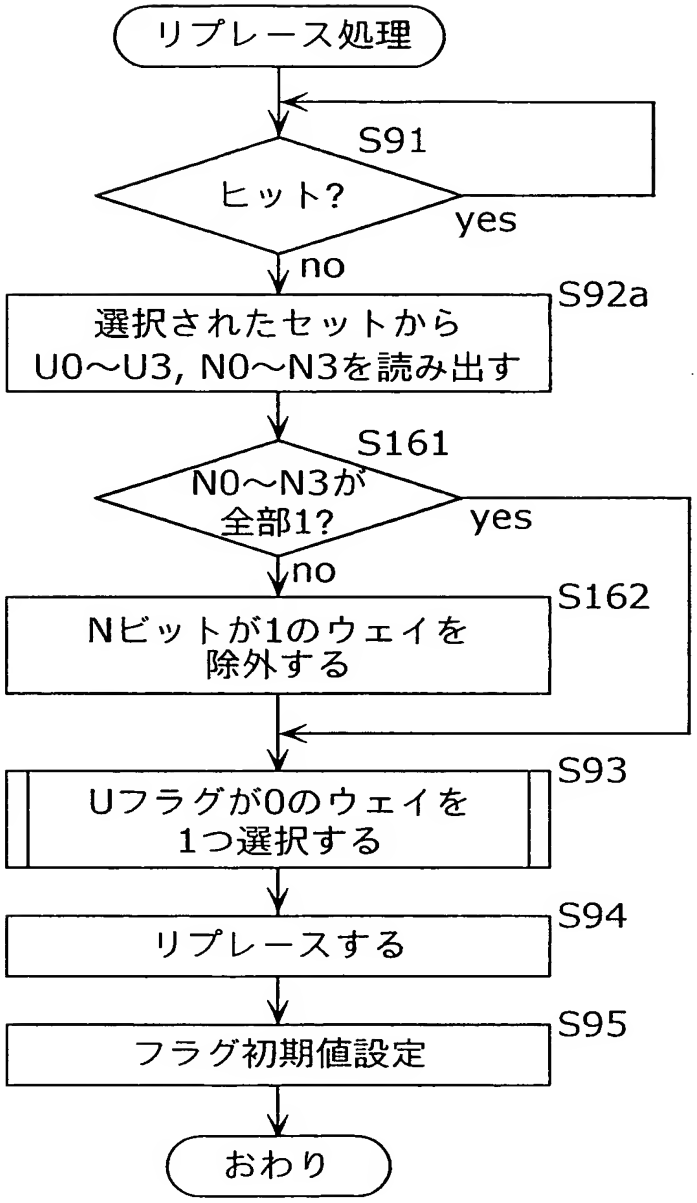

リプレース処理
S91
ヒット?
yes
no
S92a
選択されたセットから
U0～U3, N0～N3を読み出す
S161
N0～N3が
全部1?
yes
no
S162
Nビットが1のウェイを
除外する
S93
Uフラグが0のウェイを
1つ選択する
S94
リプレースする
S95
フラグ初期値設定
おわり

図14

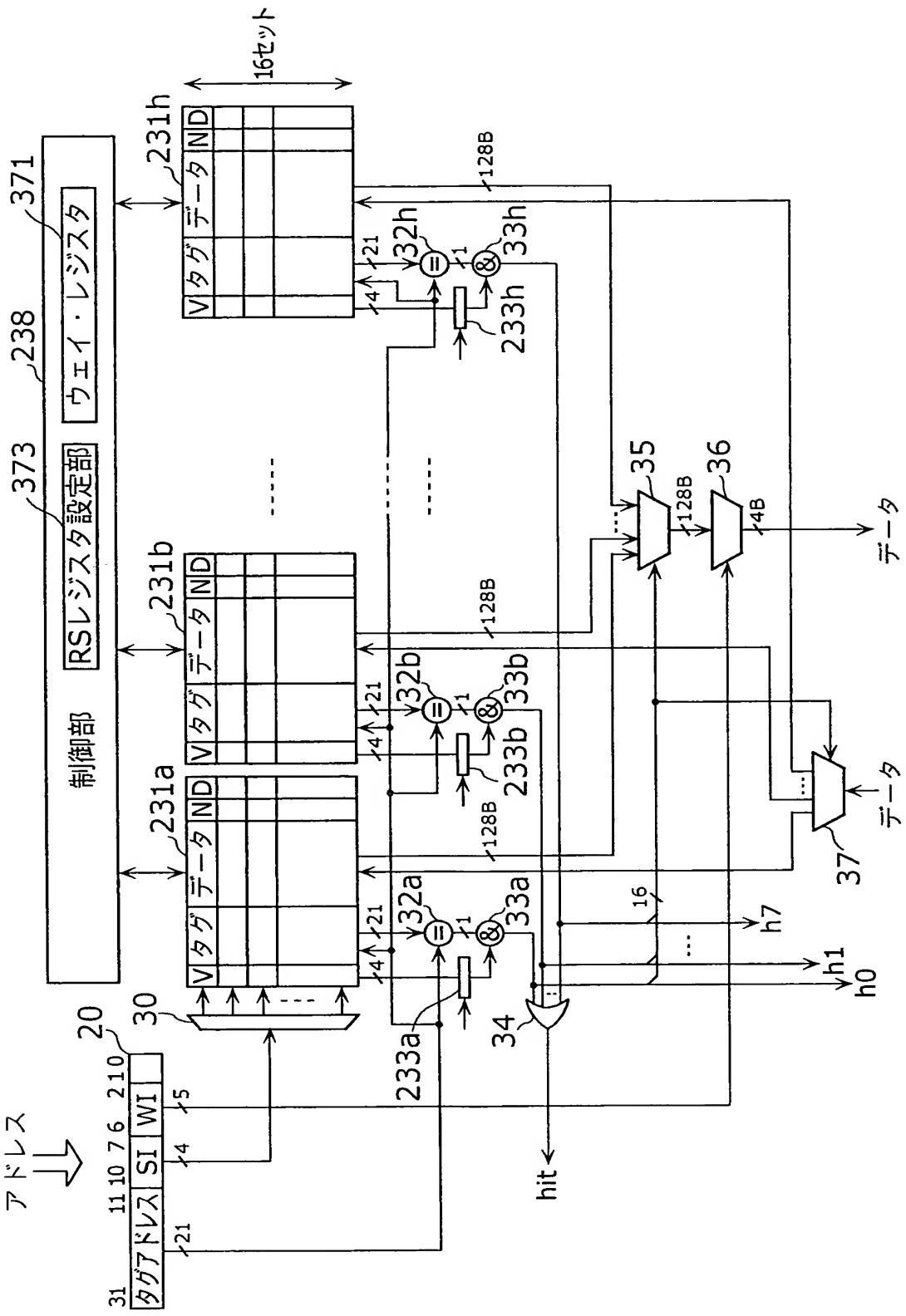

16セット
231h
231b
231a
V タグ データ ND
371
ウェイ・レジスタ
238
373
RSレジスタ設定部
制御部
20
30
34
35
36
37
32a
32b
32h
33a
33b
33h
233a
233b
233h
128B
4B
16
21
4
1
5
h7
h1
h0
hit
データ
アドレス
31 タグアドレス 11 10 7 6 SI 2 1 0 WI

図15

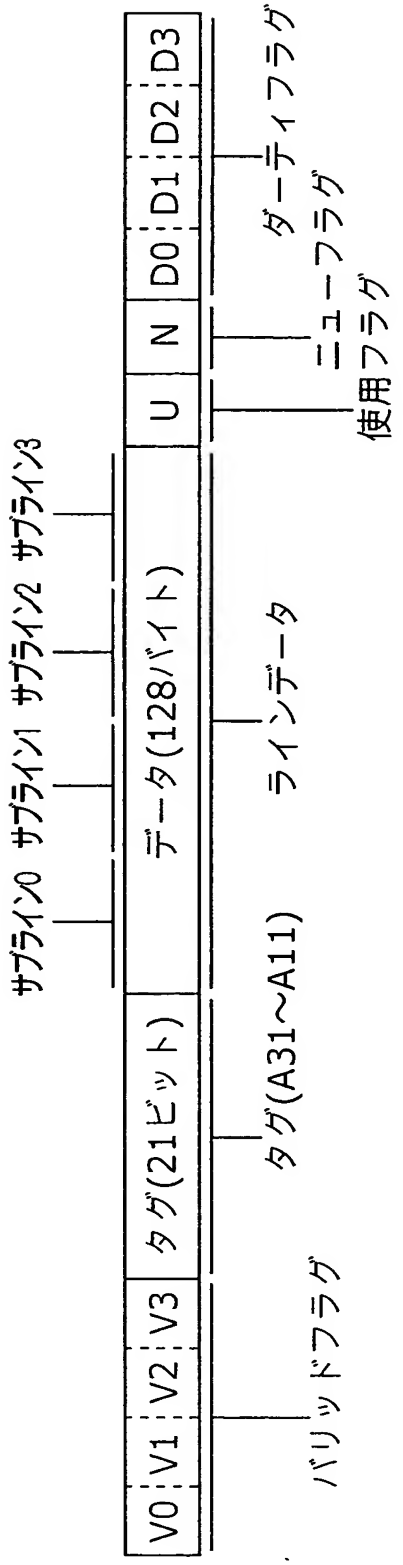

サブライン0
サブライン1
サブライン2
サブライン3
データ(128バイト)
タグ(21ビット)
V0
V1
V2
V3
U
N
D0
D1
D2
D3
パリッドフラグ
タグ(A31～A11)
ラインデータ
ニューフラグ
使用フラグ
ダーティフラグ

図16

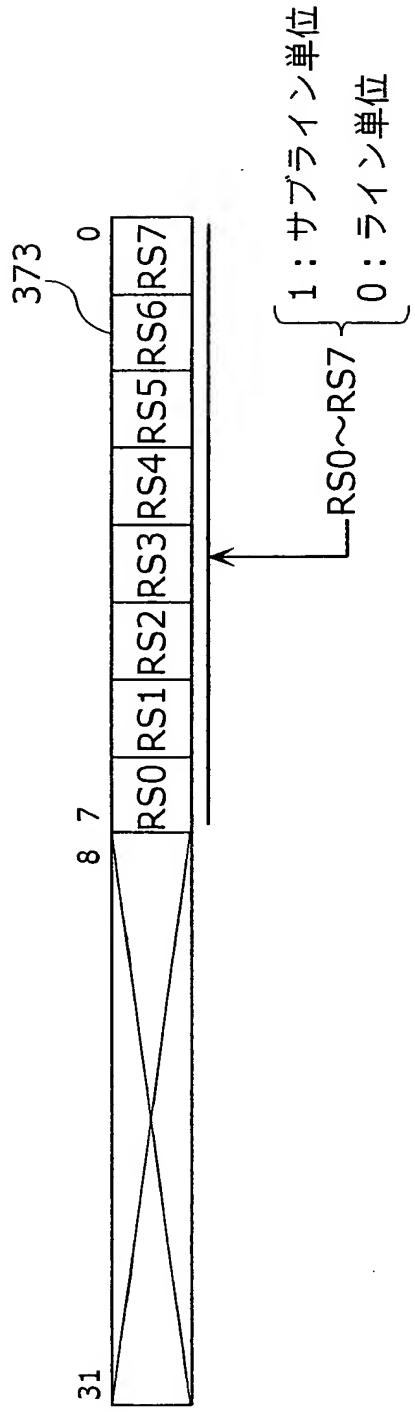

373
0
RS7
RS6
RS5
RS4
RS3
RS2
RS1
RS0
8 7
31
RS0～RS7
1：サブライン単位
0：ライン単位

図17

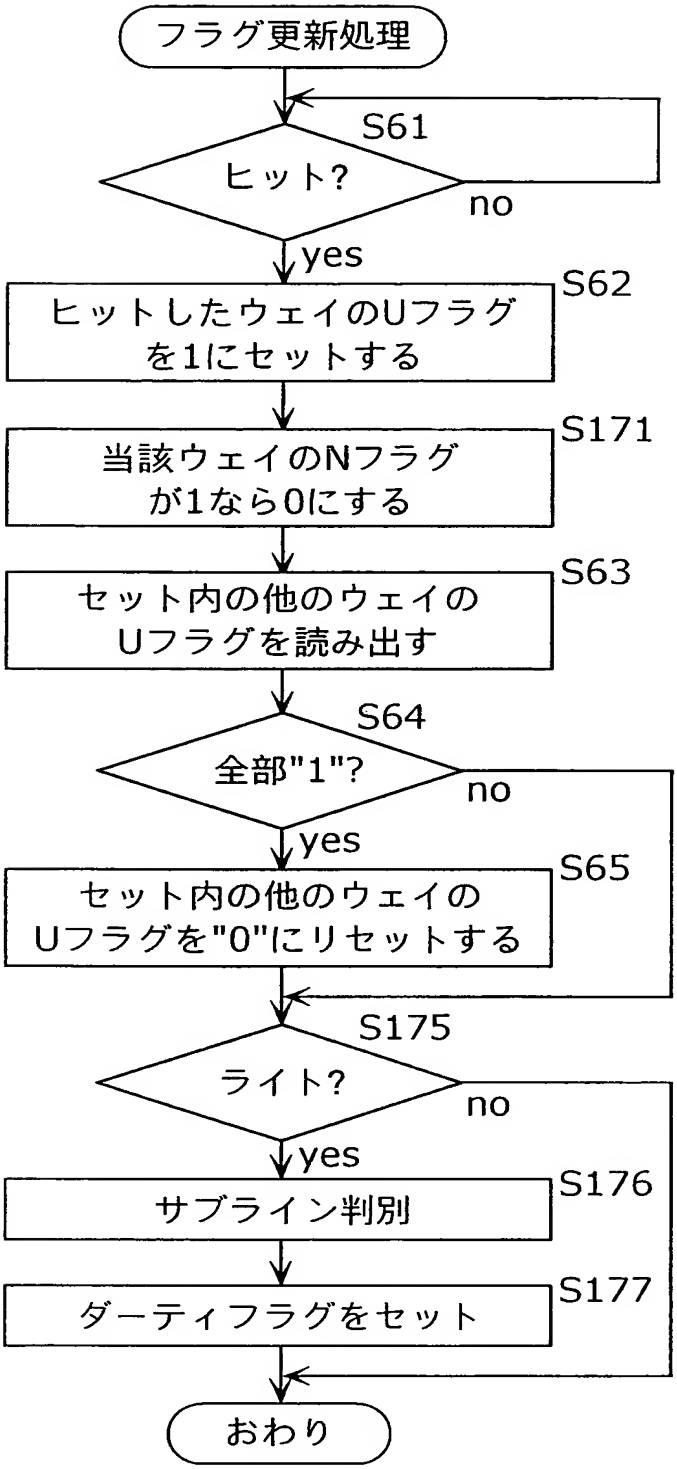

フラグ更新処理
S61
ヒット?
no
yes
S62
ヒットしたウェイのUフラグ
を1にセットする
S171
当該ウェイのNフラグ
が1なら0にする
S63
セット内の他のウェイの
Uフラグを読み出す
S64
全部"1"?
no
yes
S65
セット内の他のウェイの
Uフラグを"0"にリセットする
S175
ライト?
no
yes
S176
サブライン判別
S177
ダーティフラグをセット
おわり

図18

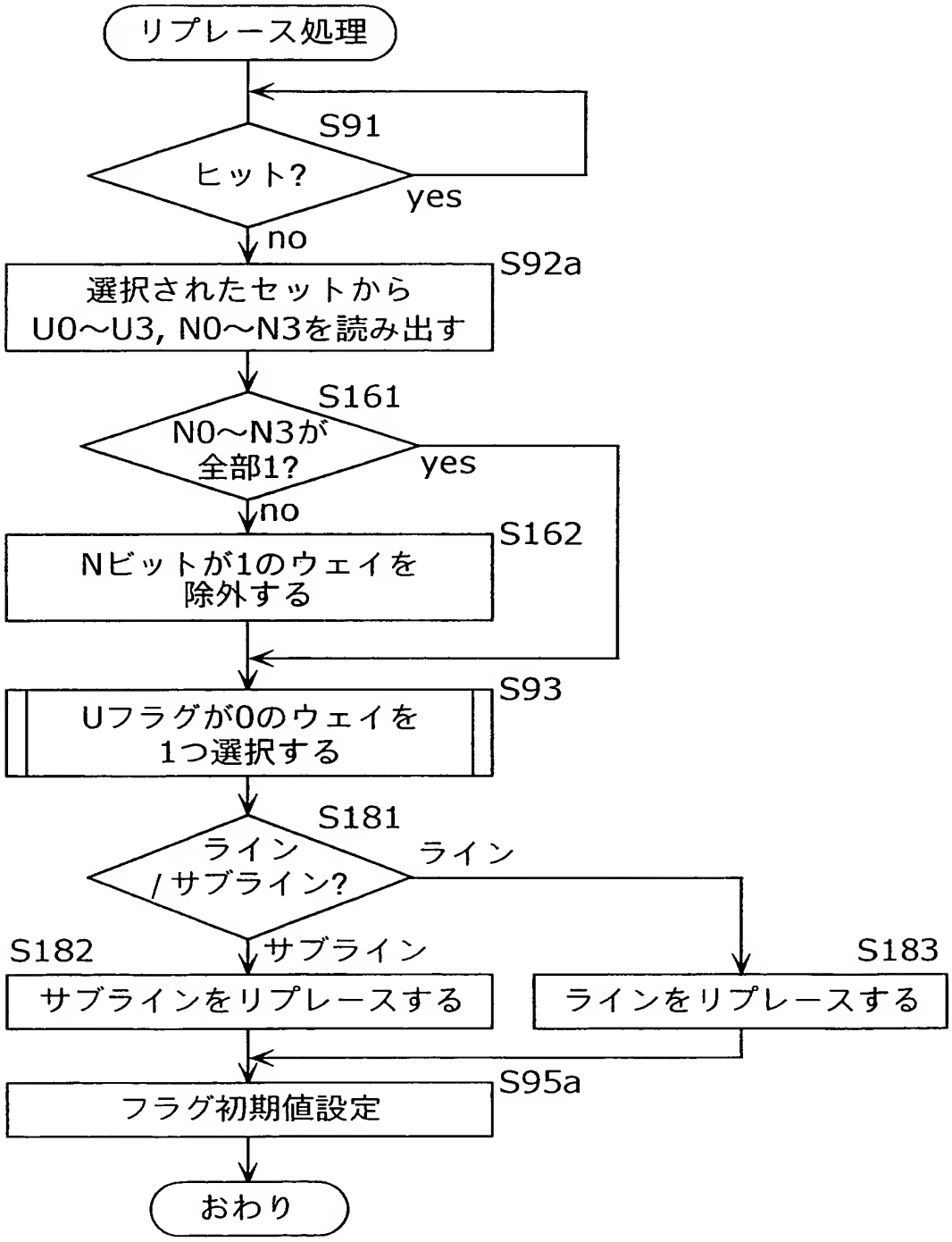

リプレース処理
S91
ヒット?
yes
no
S92a
選択されたセットから
U0～U3, N0～N3を読み出す
S161
N0～N3が
全部1?
yes
no
S162
Nビットが1のウェイを
除外する
S93
Uフラグが0のウェイを
1つ選択する
S181
ライン
/サブライン?
ライン
サブライン
S182
サブラインをリプレースする
S183
ラインをリプレースする
S95a
フラグ初期値設定
おわり

図19

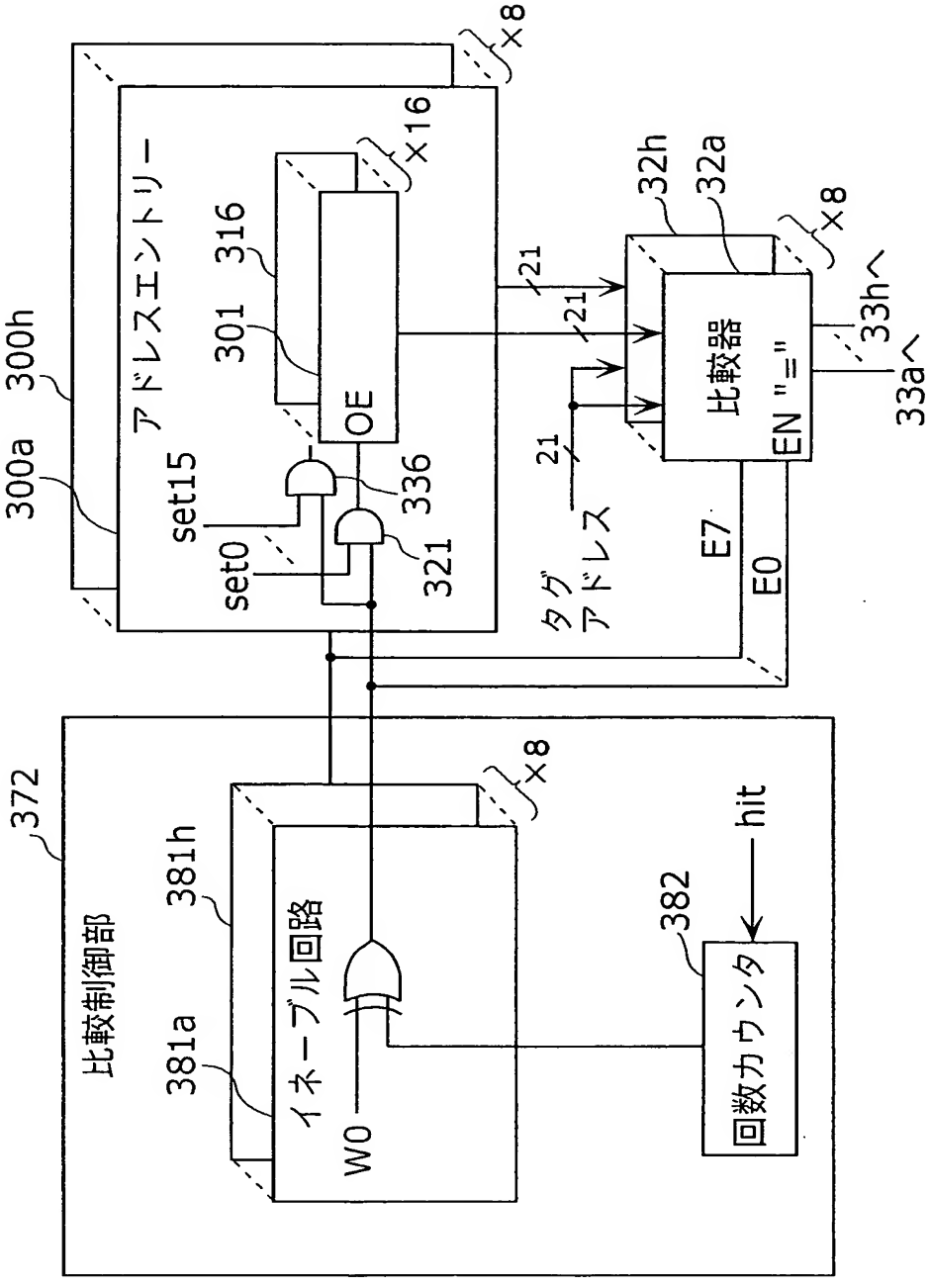

×8
×16
アドレスエントリー
300h
300a
316
301
OE
set15
set0
336
321
21
21
32h
32a
×8
比較器
EN "="
33h
33a
タグ
アドレス
E7
E0
372
比較制御部
381h
381a
イネーブル回路
W0
×8
382
回数カウンタ
hit

図20

入力		出力
Wn(n=0~7)	カウント値	En
1(アクティブ)	0(1回目)	1(イネーブル)
1(アクティブ)	1(2回目)	0(ディスエーブル)
0(インアクティブ)	0(1回目)	0(ディスエーブル)
0(インアクティブ)	1(2回目)	1(イネーブル)